

Roberto Poli
Michael Healy
Achilles Kameas
Editors

Theory and Applications of Ontology

Computer Applications

 Springer

Theory and Applications of Ontology: Computer Applications

Roberto Poli · Michael Healy · Achilles Kameas
Editors

Theory and Applications of Ontology: Computer Applications

 Springer

Editors

Roberto Poli
University of Trento
Dept. Sociology and Social Research
26, Verdi str.
38100 Trento
Italy
roberto.poli@unitn.it

Dr. Michael Healy
13544 23rd Place NE.,
Seattle WA 98125-3325
USA
mjhealy@ece.unm.edu

Achilles Kameas
Hellenic Open University & Computer
Technology Institute
N. Kazantzaki Str.
265 00 Patras
University Campus
Greece
kameas@cti.gr; ie2006@cti.gr;
kameas@eap.gr

ISBN 978-90-481-8846-8 e-ISBN 978-90-481-8847-5
DOI 10.1007/978-90-481-8847-5
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2010933121

© Springer Science+Business Media B.V. 2010

Chapter 11 is published with kind permission of © All rights reserved

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

After a long period of decline, ontology is back at the forefront of philosophy, science and technology. These days ontology comes in at least two main fashions: the traditional philosophical understanding of ontology has been recently flanked by a new – computer-based – understanding of ontology.

There are scholars from both fields contending that ontology in knowledge engineering and ontology in philosophy are two completely different disciplines. On the one hand there is analysis closely tied to the concrete problems of domain modeling; on the other, difficult and usually very abstract speculations on the world and its most rarified structures. For this reason, it is claimed, those scientists who occupy themselves with ontology in knowledge engineering should not be concerned with what philosophers have to say (and vice-versa).

The thesis defended by *Theory and Applications of Ontology* is exactly the opposite. We shall try to show in this work that – despite their different languages and different points of departure – ontologies in knowledge engineering (let's say: ontology as technology) and ontology in philosophy (let's say: ontology as categorial analysis) have numerous problems in common and that they seek to answer similar questions. And for this reason, engineers and philosophers must devise ways to talk to each other.

The current resurgence of interest in ontological issues displays a number of novel features, both among philosophers and among information technologists. Among philosophers, the revival of a genuine interest in ontology requires the removal of certain prejudices that have profoundly influenced the analytic and the continental camps, both of which have in recent decades systematically delegitimized ontological inquiry in favour of its epistemological transformation (not to say reduction). To this shared error of broadly Kantian (or more properly neo-Kantian) stamp, analytic philosophy has added a linguistic prejudice, and the continental one styles of inquiry and writing that can be described as devoid of methodological rigour.

Behind these obstructions to ontological investigation one perhaps discerns the consequences of another feature common to both camps: the fact that the most influential thinkers of the last hundred years – the reference unquestionably goes back to Wittgenstein and Heidegger, however different their philosophical views may have been – both embraced an a-scientific approach; both, that is, delegitimized alliances,

or at least serious contact, between science and philosophy. In consequence, the revival of interest in ontology also provides an opportunity for renewed discussion of the relationships between science and philosophy.

Science continuously advances, and that which it proves to be valid endures. Problem-oriented thinkers try to follow problems, not to anticipate conclusions or to presuppose an image of the world. This perspective is largely correct. It should, however, be qualified if one is not to commit the ingenuous error of believing that it is only “solutions” that advance knowledge. Also attempts and failures, in fact, are instructive. For all these reasons we may accept Aristotle’s contention that ontology is *philosophia prima* as regards the problems it seeks to resolve, as long as we remember that it can only be *philosophia ultima* as regards the elaboration of results. And it is here that we discern how ontology concretely operates in harness with science, because it “presupposes the accumulated knowledge of centuries and the methodical experience of all the sciences” (N. Hartmann, *Der Aufbau der realen Welt*, Meisenheim am Glan, 1949, 26).

Besides points of contact, of course, there are also a number of differences, perhaps most notably the fact that ontology in knowledge engineering is a discipline still in its infancy, while ontology in philosophy is as old as philosophy itself. Consequently, the history of philosophy contains ideas, tools and proposals of use for contemporary developments; and it also indicates the options that will lead us into dead ends or nowhere at all. When things are viewed in the light of such a long and articulated history, one knows from the outset that ontology does not permit ingenuous simplifications. For these reasons, philosophical ontology may usefully contribute to ontology in knowledge engineering.

It is true, though, that philosophical ontology addresses questions of a more general nature, ones apparently of no relevance to ontology in knowledge engineering. Consequently, it may appear that certain components of philosophical ontology could be ignored in the passage to ontology as technology. Nevertheless, one should always bear in mind the greater explanatory value and the broader structuring capacity of more general schemes and more comprehensive theories. For this less overt reason, too, philosophical ontology is useful for ontology in knowledge engineering.

The philosophical codification of ontology has often restricted itself to organization of its general architecture, without delving into the details of minute categorization. On the other hand, the concrete, situated practice of ontology as technology may conversely prove useful for the development of philosophical ontology.

For these and other reasons, there is mounting interest in the development of standards, modeling principles, and semantically transparent languages. Ontology thus comes into play as one of the strategies available to developing the semantic web, construct robust data-bases, managing huge amounts of heterogeneous information because ontologically founded knowledge of the objects of the world is able to make codification simpler, more transparent and more natural. The belief is that ontology can give greater robustness to computer-based applications by providing methodological criteria and categories with which to construct and build them, as

well as contexts in which to set and re-categorize different data-bases so that they become more mutually transparent. In this way ontology directly contributes to standardization of the life-cycle model, and can therefore serve as an innovative and possibly unexpected component of software quality assurance.

These problems are dramatically magnified by the fact that unlike all the societies of the past, modern societies are no longer afflicted by a lack of information. If anything they suffer from its excess, from having to cope with too much unused and unusable information. It becomes increasingly difficult, in fact, to find the information that one needs, when one needs it, to the extent that one needs it and in the appropriate form. Although the information may be stored somewhere, all too often one does not know where; and even when one is aware of how to find the information, it is often accompanied by further information irrelevant to one's purposes. And when information is available, it is often forthcoming in the wrong form, or else its meaning is not explicitly apparent.

However broad the range of information already gathered may be, a great deal more has still to be assembled and codified. And this inevitably complicates still further the problem of the functional, flexible, efficient and semantically transparent codification of information.

Broadly speaking, the two research communities of philosophers and engineers have still not found a way to relate to each other systematically. While philosophers tend unilaterally to emphasize the need for a conceptual complexity that matches the complexity of the subject-matter, engineers tend equally unilaterally to stress the drawbacks of the tools available and the presence of insuperable computational problems. One side is perhaps too theoretical, the other too pragmatic. In short, taken as they stand, the two views seem difficult to reconcile.

However, in dynamic terms, one easily foresees mounting social and institutional pressure for the development of tools able to model fragments of reality in terms that are both adequate and efficient. And from this point of view, we are all at fault. Those colleagues who concern themselves with technologies seemingly pay closer attention to manipulation than to knowledge. Likewise, those who concern themselves with philosophy suffer from the reverse problem, that of navigating in a sea of theories for which the rationale is sometimes unclear.

For our part, we have grown increasingly convinced that the same problems will force engineers to address theories, and philosophers to address the limitations of our current capabilities. Provided, however, that both sides have the will, the ability, the desire and the courage to do so. If they decide to tackle these problems, it will become reasonable to identify and systematically develop those areas of convergence and contact now existing.

In this sense, the two volumes of *Theory and Applications of Ontology* may play a role in paving the way for a better mutual understanding between engineers and philosophers. Since the two communities are still very different as to their own languages, conceptual tools and problem-sets, we thought that collecting papers within one single volume would have been too constraining. We therefore devised two

different volumes, one dedicated to the philosophical understanding of ontology and one to the computer-based understanding of ontologies. Both volumes contain both papers describing the state of the art in their respective topics and papers addressing forefront, innovative and possibly controversial topics.

Roberto Poli

Contents

1	The Interplay Between Ontology as Categorical Analysis and Ontology as Technology	1
	Roberto Poli and Leo Obrst	
2	Ontological Architectures	27
	Leo Obrst	
3	Organization and Management of Large Categorical Systems	67
	Frank Loebe	
4	The Information Flow Approach to Ontology-Based Semantic Alignment	101
	Yannis Kalfoglou and Marco Schorlemmer	
5	Ontological Evaluation and Validation	115
	Samir Tartir, I. Budak Arpinar, and Amit P. Sheth	
6	Tools for Ontology Engineering and Management	131
	Lambrini Seremeti and Achilles Kameas	
7	Ontological Tools: Requirements, Design Issues and Perspectives . .	155
	Konstantinos Kotis and George Vouros	
8	Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages	175
	Giancarlo Guizzardi and Gerd Wagner	
9	Lightweight Ontologies	197
	John Davies	
10	WordNet	231
	Christiane Fellbaum	
11	Controlled English to Logic Translation	245
	Adam Pease and John Li	
12	Cyc	259
	Douglas Foxvog	

13	Ontological Foundations of DOLCE	279
	Stefano Borgo and Claudio Masolo	
14	General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling	297
	Heinrich Herre	
15	Ontologies in Biology	347
	Janet Kelso, Robert Hoehndorf, and Kay Prüfer	
16	The Ontology of Medical Terminological Systems: Towards the Next Generation of Medical Ontologies	373
	Heinrich Herre	
17	Ontologies of Language and Language Processing	393
	John A. Bateman	
18	Business Ontologies	411
	Peter Rittgen	
19	Ontologies for E-government	429
	Knut Hinkelmann, Barbara Thönssen, and Daniela Wolff	
20	An Ontology-Driven Approach and a Context Management Framework for Ubiquitous Computing Applications	463
	Christos Goumopoulos and Achilles Kameas	
21	Category Theory as a Mathematics for Formalizing Ontologies . .	487
	Michael John Healy	
22	Issues of Logic, Algebra and Topology in Ontology	511
	Steven Vickers	
23	The Institutional Approach	533
	Robert E. Kent	
24	Ontology Engineering, Universal Algebra, and Category Theory .	565
	Michael Johnson and Robert Rosebrugh	

Contributors

I. Budak Arpinar Department of Computer Science, UGA, Athens GA, 30602-7404, USA, budak@cs.uga.edu

John A. Bateman University of Bremen, Bremen, Germany, bateman@uni-bremen.de

Stefano Borgo Laboratory for Applied Ontology, National Research Council, Institute of Cognitive Sciences and Technologies – ISTC–CNR; KRDB, Free University of Bolzano, Via alla Cascata, 56C, 38123 Povo-Trento (TN), Italy, borgo@loa-cnr.it

John Davies BT Innovate, British Telecommunications plc, London, UK, john.nj.davies@bt.com

Christiane Fellbaum Department of Computer Science, Princeton University, Princeton, NJ 08540, USA, fellbaum@princeton.edu

Douglas Foxvog Digital Enterprise Research Institute (DERI) Ireland, Galway, Ireland, doug.foxvog@deri.org

Christos Goumopoulos Distributed Ambient Information Systems Group, Hellenic Open University & Computer Technology Institute, Patras, Hellas, Greece, goumop@cti.gr

Giancarlo Guizzardi Federal University of Espirito Santo (UFES), Vitoria-ES, Brazil; Laboratory for Applied Ontology (ISTC-CNR), Trento, Italy, gguizzardi@inf.ufes.br

Michael John Healy Department of Electrical and Computer Engineering, University of New Mexico, Seattle, WA 98125, USA, mjhealy@ece.unm.edu

Heinrich Herre Research Group Onto-Med, IMISE, University Leipzig, Leipzig, Germany, heinrich.herre@imise.uni-leipzig.de

Knut Hinkelmann University of Applied Sciences Northwestern Switzerland FHNW, Olten, Switzerland, knut.hinkelmann@fhnw.ch

Robert Hoehndorf Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, Leipzig 04103, Germany; Department of Computer Science, University of Leipzig, Leipzig, Germany, leechuck@leechuck.de

Michael Johnson Department of Computer Science, Macquarie University, Sydney, Australia, mike@ics.mq.edu.au

Yannis Kalfoglou Ricoh Europe Plc, London, UK, ykalfoglou@googlemail.com

Achilles Kameas Distributed Ambient Information Systems Group, Hellenic Open University & Computer Technology Institute, Patras, Hellas, Greece, kameas@eap.gr

Janet Kelso Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, Leipzig 04103, Germany, kelso@eva.mpg.de

Robert E. Kent Ontologos, Pullman, Washington, USA, re Kent@ontologos.org

Konstantinos Kotis AI Lab, Department of Information and Communications Systems Engineering, University of the Aegean, 83200 Karlovassi, Greece, kotis@aegean.gr

John Li Articulate Software, Mountain View, CA, USA, jli@articulatesoftware.com

Frank Loebe Department of Computer Science, University of Leipzig, Leipzig, Germany, frank.loebe@informatik.uni-leipzig.de

Claudio Masolo Laboratory for Applied Ontology, National Research Council, Institute of Cognitive Sciences and Technologies – ISTC–CNR, Via alla Cascata, 56C, 38123 Povo-Trento (TN), Italy, masolo@loa-cnr.it

Leo Obrst The MITRE Corporation, McLean, VA, USA, lobrst@mitre.org

Adam Pease Articulate Software, Mountain View, CA, USA, apease@articulatesoftware.com

Roberto Poli University of Trento, Trento, Italy, Roberto.Poli@unitn.it

Kay Prüfer Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, Leipzig 04103, Germany, pruefer@eva.mpg.de

Peter Rittgen Vlerick Leuven Gent Management School, Leuven, Belgium; University of Borås, Borås, Sweden, Peter.Rittgen@vlerick.com

Robert Rosebrugh Department of Mathematics and Computer Science, Mount Allison University, Sackville, Canada, rrosebrugh@mta.ca

Marco Schorlemmer Artificial Intelligence Research Institute (III A), CSIC, Bellaterra (Barcelona), Catalonia, Spain, marco@iiia.csic.es

Lambrini Seremeti Distributed Ambient Information Systems Group, Hellenic Open University & Computer Technology Institute, Patras, Hellas Greece, seremeti@cti.gr

Amit P. Sheth Kno.e.sis Center, Department of Computer Science & Engineering, Wright State University, Dayton, Ohio 45435-0001, USA, amit.sheth@wright.edu

Samir Tartir Faculty of Information Technology, Philadelphia University, Amman 19392, Jordan, startir@philadelphia.edu.jo

Barbara Thönssen University of Applied Sciences Northwestern Switzerland FHNW, Olten, Switzerland, barbara.thoenssen@fhnw.ch

Steven Vickers School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK, s.j.vickers@cs.bham.ac.uk

George Vouros AI Lab, Department of Information and Communications Systems Engineering, University of the Aegean, 83200 Karlovassi, Greece, georgev@aegean.gr

Gerd Wagner Brandenburg University of Technology at Cottbus, Cottbus, Germany, wagnerg@tu-cottbus.de

Daniela Wolff University of Applied Sciences Northwestern Switzerland FHNW, Olten, Switzerland, daniela.wolff@fhnw.ch

Introduction

Recent events in information technology have led to a new manifestation of the philosophical field of ontology. In this new manifestation, ontology is also a technological discipline. On reflection, this development can be seen as unsurprising: Ontology arises naturally in investigations of advanced information processing systems such as knowledge-based systems and the world-wide web. The development of knowledge-based systems has led to computer applications written to manage knowledge expressed in symbolic form, in a variety of domains such as diagnostics and manufacturing engineering and in a variety of programming languages. Each system has its own set of engineering or medical or scientific artifacts for different domains of knowledge, its own rules expressing domain relationships, and its own terminology. This makes interoperability difficult if not intractable. The philosophical notion of ontology suggests a possible solution in the form of a system-neutral repository of abstract knowledge which can be refined to specify system rules and artifacts in the domains to be modeled, accompanied by automated translators mediating between each knowledge system and the repository. Another example concerns the semantic web, a proposed new-generation world-wide web meant to achieve a deeper, more meaningful communication between users, their browsers and the web sites they access than is possible with the purely syntactic medium of key words and “icons”. But this begs the question: What can one communicate when there is no common basis for meaning? Again, by appropriating the philosophical notion of ontology, technologists hope to resolve the underlying issues of meaning and communication among users, systems, and content.

The two volumes of Theory and Applications of Ontology (TAO) are intended to inform the scholar in philosophy or the researcher in the sciences, information technology, or engineering, of the present state of the art in philosophical ontology and the systems available for the study, development, and application of ontology as technology. While Volume 1 addresses philosophy, the present volume, Volume 2, addresses the recent flowering of ontology as an all-encompassing field of study and application, which provides a declarative semantic framework for mutual understanding and interoperability between technological system components, models and processes. Volume 2 is intended as a snapshot of much, although not all, of the work in progress on ontology in this new role as a component of technological systems.

The chapters in this second volume of TAO are grouped in four parts. We consider this grouping necessary, in order to help the reader deal with the large volume of knowledge contained in the book. Of course, this grouping does not mean that the chapters are not related or interrelated; in fact, the reader will discover references from chapters that present seemingly different aspects of ontologies to common concepts and entities, which constitutes a proof of the universal application of ontologies. The chapters in the first part of the book support this claim. The chapters in the second and third parts, which constitute the largest part of the book, present the application of ontologies to specific domains, thus justifying the sub-title of the volume at-hand. We do not aim to provide an exhaustive catalogue of ontologies available, but to help the reader in forming the necessary cognitive structures that will allow him to classify ontology applications and evaluate correctly the tools and methodologies. The final part contributes chapters that shed light into the formalisms used to describe and manipulate ontologies, closing in a way the path that started with the chapters in Volume 1 of this set. Nevertheless, each of the two volumes is self-contained and can be studied independently.

As we already mentioned, the first part in Volume 2 contains the chapters that provide an overview of various perspectives of ontology theory, architecture, constructs and application. In this context, Poli and Obrst present an overview of ontology from both the philosophical and technological perspectives. This is by way of introducing Volume 2, the assumption being that this discussion would be unnecessary were the philosophical view the only one to be represented in these volumes. Continuing, Obrst distinguishes between *ontology architecture*, as a distinct discipline in ontology engineering, which includes ontology lifecycle management, and *ontological architecture* as the architecture used to structure ontologies. The latter addresses both ontological levels (foundational, upper, middle, utility, reference, domain, and sub-domain ontologies), and formal constructs used to modularize ontologies in a large *ontological space*. Loebe surveys approaches to handling categorical systems of extensive size, spanning from semi-formal systems in terminology and classification sciences to formal logical approaches. In particular, he reviews the transition from terminologies to ontologies that are formalized in logics, exemplified in the medical domain. Tartir, Arpinar, and Sheth introduce several approaches that have been developed to aid in evaluating ontologies. As part of this, they present highlights of OntoQA, an ontology evaluation and analysis tool that uses a set of metrics measuring different aspects of an ontology's schema and knowledge base. Seremeti and Kameas provide an overview of the available tools and software environments that can be used for building, maintaining, and evolving ontologies. Kotis and Vouros aim to provide an understanding of the functionalities and technologies that need to be integrated in ontology engineering environments by presenting issues that next generation ontological tools must consider.

The second part groups the chapters that discuss specific ontologies, foundational ontologies and ontology engineering systems. Guizzardi and Wagner present the Unified Foundational Ontology (UFO), which was developed to serve as a foundation for general conceptual modeling languages. They demonstrate the use of

this foundational ontology in the evaluation and redesign of the Unified Modeling Language (UML) for concept-based software development. Pease and Li introduce the Controlled English to Logic (CELT) system, which translates a restricted English grammar to expressions in formal logic. The logic statements use terms from a large formal ontology, the Suggested Upper Merged Ontology (SUMO). Foxvog presents the Cyc system familiar to AI researchers. The original intent of the Cyc project, begun in the 1980s, was to produce an ontology of “all commonsense knowledge.” Borgo and Masolo present the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), a foundational ontology embracing the notion that there cannot be a unique standard or universal ontology for knowledge representation. Herre has two contributions in this volume. In the first chapter, he presents the General Formal Ontology (GFO), a foundational ontology for conceptual modelling.

The third part presents the application of ontologies in different disciplines as components used to provide semantically rich representations of technological domains. The contributions in this part discuss specific domain ontologies as well as issues in the engineering of ontologies for specific domains. Davies and Kiryakov distinguish *lightweight ontologies* (which roughly correspond to ontologies for application domains in information technology) from the philosophical notion of ontologies and then motivate and describe techniques for translating information modelling schemes into lightweight ontologies.

Natural language (English, German, etc.), usually in some simplified, semi-formalized form making it amenable to computer processing, is often the basis either for expressing ontologies or as a domain for applications of ontology. Bateman discusses approaches to natural language processing where there is a strong interaction with ontological engineering. Fellbaum presents WordNet, a large electronic lexical database for English. WordNet is in fact a semantic network expressing relationships between words with similar meaning and, hence, has become a valuable tool for Natural Language Processing and has spawned research in lexical semantics and ontology.

In his second chapter, Herre applies the GFO as an analysis and development tool for biomedical ontologies. Kelso, Hoehndorf and Prulfer more generally address ontologies for the biomedical domain. They discuss the formalization of community knowledge in molecular biology along with the provision of a shared vocabulary for the annotation of the growing amount of biological data available. Rittgen discusses a number of approaches, rooted in different fields of research, to modeling the multifaceted business domain. The emphasis is placed upon pragmatism in modeling enterprise ontologies. Feldkamp, Hinkelmann, and Thoenssen discuss ontologies for e-government. There are issues here similar to those for business, e.g., minimizing the cost of government, except that public service at state, provincial and municipal levels are involved and also regulation of commerce and other activities of businesses and other public/private entities. Goumopoulos and Kameas present an ontology-driven approach and a context management framework for composing context-aware ubiquitous computing applications. The focus is upon applications which combine the services offered by heterogeneous everyday physical objects

(e.g., information devices, home appliances, etc) that have been enhanced with sensing, acting, processing and communication abilities.

The fourth and final part of the current volume compiles chapters that bring the notions of ontology formalization and formal ontologies into the realm of mathematical rigor. Healy posits that an appropriate mathematical language for this is category theory, the mathematics of structure. The ensuing discussion introduces this field, which is also referred to as conceptual mathematics, and proceeds from basic-level definitions and explanations to an in-depth exposition of some of its key notions. This serves to introduce the chapters by Kalfoglou and Schorlemmer, Vickers, Kent, and Johnson and Rosebrugh. Each of these chapters approaches the subject of ontology in a different way, yielding an indication of the richness of category theory as conceptual mathematics. Kalfoglou and Schorlemmer discuss the semantic alignment of ontologies as systems of categorical relationships called information systems, which show how the terms in the different ontologies are associated. Vickers discusses the ontological commitments made by a form of categorical logic which has been called the logic of observable quantities; this logic is well-adapted to formalizing ontologies for scientific theories. Kent provides an exposition of work for ontologies over the World-Wide Web that is based upon the work of Joseph Goguen on institutions. The latter are mathematical systems for analyzing and clarifying the semantics of different logics. Johnson and Rosebrugh provide a general scheme for the use of category theory in ontology by presenting a category-theoretic approach to ontology engineering.

The four parts of this volume aim at providing comprehensive coverage of the current uses of ontologies as components of technological systems. They have been structured in a way that guides the reader from overviews of ontology application to mathematical formalization, passing through ontology engineering systems and domain-specific ontologies. This structure reflects the editors' choice of most profitable studying path. However, each part is independent from the others and will equip the reader with updated and complete knowledge under a specific perspective in ontology engineering and application. The reader is advised to study one part thoroughly before moving to the next, as the chapters in each part complement each other under the part's perspective. Each chapter has been authored by distinguished scholars in the various applications of ontologies. Let them guide you, the reader, in a path of knowledge discovery that we, the volume editors, find to be the most fascinating.

Chapter 1

The Interplay Between Ontology as Categorial Analysis and Ontology as Technology

Roberto Poli and Leo Obrst

1.1 Introduction

The notion of ontology today comes with two perspectives: one traditionally from philosophy and one more recently from computer science. The philosophical perspective of ontology focuses on categorial analysis, i.e., what are the entities of the world and what are the categories of entities? *Prima facie*, the intention of categorial analysis is to inventory reality. The computer science perspective of ontology, i.e., ontology as technology, focuses on those same questions but the intention is distinct: to create engineering models of reality, artifacts which can be used by software, and perhaps directly interpreted and reasoned over by special software called inference engines, to imbue software with human level semantics. Philosophical ontology arguably begins with the Greek philosophers, more than 2,400 years ago. Computational ontology (sometimes called “ontological” or “ontology” engineering) began about 15 years ago.

In this chapter, we will focus on the interaction between ontology as categorial analysis (“ontology_c”, sometimes called “Big O” ontology) and ontology as technology (“ontology_t”, sometimes called “Little o” ontology). The individual perspectives have each much to offer the other. But their interplay is even more interesting.¹

This chapter is structured in the following way. Primarily we discuss ontology_c and ontology_t, introducing notions of both as part of the discussion about their interplay. We don’t think they are radically distinct and so do not want to radically distinguish them, intending by the discussion of the interplay to highlight their distinctions where they occur, but thereby emphasize their correspondences, and, in fact, their correlations, complementarities, interdependencies. They are distinct

R. Poli (✉)
University of Trento, Trento, Italy
e-mail: Roberto.Poli@unitn.it

¹Cf. Daconta et al. (2003, p. 186). The first use of this “Big O, little o” terminology, as known by the authors, is in Guarino (1995). The distinction made between ontology_c and ontology_t is first made in Poli (2001b).

perspectives after all, we want to emphasize, not distinct analytical methodologies, nor do they provide distinct analytical products. We discuss some of the historical definitions of ontology_t, as they emerged during the 1990s. We then provide our own take on the nature of ontology_t. As part of this exposition, we briefly discuss the levels and representation of ontologies, ranging over the typical levels of upper ontologies, middle ontologies, and domain (and sub-domain) ontologies.

Although we cannot discuss the knowledge representation languages typically used by ontology_t, from Semantic Web languages such as OWL² (primarily a description logic) to First-Order Logic (FOL, predicate calculus) languages such as ISO Common Logic,³ nor the automated reasoning over ontologies that is of potential benefit to ontology_c as well as to ontology_t, we consider these issues important but better exposed in another venue. The interested reader is, therefore, directed to [Chapter 2, Ontology Architecture](#), for a fuller exposition.

We do, however, lay down some principles by which we believe ontologies_t should be developed, based on analysis from ontology_c, and introduce the notion of “levels of reality”. We illustrate the interplay of the two notions of ontology by providing an extended discussion of ontological entities in a hypothetical biology ontology.

Finally, we conclude by looking to the increasing interaction between these two aspects of ontology in the future. We briefly discuss some common problems which require the interplay of ontology_c and ontology_t, and which will assume much greater prominence once the more basic issues are elaborated on and scientific consensus established, i.e., ontology modularity, mapping, context-determination and representation, and vagueness and uncertainty. Ontology_t needs to be informed by ontology_c and its analytical methods. Ontology_c will increasingly benefit from the sound and consistent software engineering products arising from ontology_t.

1.2 Ontology_c

Ontology and ontologies have been given many different definitions, both on the philosophical side and on the technological side.

From the perspective of categorial analysis in philosophy, ontology has been viewed as both a part of metaphysics and as a part of science. Historically, ontology has been a branch of metaphysics, interested in formulating answers to the question of what exists, i.e., what’s the inventory of reality, and consequently in defining categories (kinds) of entities and the relationships among the categories. Metaphysics asks different questions than does ontology, notably the question about the nature of being as a whole.

In our understanding, ontology should also be viewed as following along the same path as science, i.e., that ontology organizes and classifies the results from that which science discovers about reality. Furthermore, ontology not only depends

²Bechhofer et al. (2004).

³ISO Common Logic: Common Logic Standard. <http://cl.tamu.edu/>.

on science but can also provide tools for the clarification of science itself, in the form of ontologically clarified and reconstructed sciences. Ontology and science can therefore support one another.

A further point of contention or at least confusion is that between ontology and epistemology, i.e., on the study of what is vs. the study of what is ascertained and how it is ascertained. Ontology requires knowledge about what is, and if knowledge is described as, for example, justified belief, then ontology may be thought to devolve to knowledge and from thence to belief and justification for belief, i.e., the realm of evidence, manners and methods by which one adjudicates evidence to form belief, and thus epistemology.

Ontology is not epistemology, but has a complex relationship to epistemology. Ontology is primarily about the entities, relations, and properties of the world, the categories of things. Epistemology is about the perceived and belief-attributed entities, relations, and properties of the world, i.e., ways of knowing or ascertaining things. So epistemology is about empirical evidence gleaned that will be described or characterized by ontology.

Contemporary ontology can be characterized in a number of ways, all of which can be considered layers of theory (Poli, 2003):

- (1) *Descriptive ontology* concerns the collection of *prima facie* information either in some specific domain of analysis or in general.
- (2) *Formal ontology* distills, filters, codifies and organizes the results of descriptive ontology (in either its local or global setting). According to this interpretation, formal ontology is formal in the sense used by Husserl in his *Logical Investigations* (Husserl, 2001; originally 1900–1901). Being “formal” in such a sense means dealing with categories like *thing*, *process*, *matter*, *form*, *whole*, *part*, and *number*. These are pure categories that characterize aspects or types of reality and still *have nothing to do with the use of any specific formalism*.
- (3) *Formalized ontology*: Formal codification in the strict sense is undertaken at this third level of theory construction. The task here is to find the proper formal codification for the constructs descriptively acquired and categorially purified in the way just indicated. The level of formalized constructions also relates to evaluation of the adequacy (expressive, computational, cognitive) of the various formalisms, and to the problem of their reciprocal translations. In this sense, formalized ontology refers to the actual formalization of ontology in a logical language, typically but not always First Order Logic (FOL). In ontology_t, this could be rendered in a knowledge representation language such as the FOL-based ISO Common Logic or in the description logic-based Web Ontology Language OWL.

The close similarity between the terms “formal” and “formalized” is rather unfortunate. One way to avoid the clash is to use “categorical” instead of “formal”.⁴

⁴Note the philosophical, common use of “categorical” instead of the term “categorial” employed in this chapter, which comes closer however to the mathematician and logician’s use of the term “categorial”, as for example in Category and Topos Theory.

Most contemporary theory recognizes only two levels of work in ontology and often merges the level of the formal categories either with that of descriptive or with that of formalized analysis. As a consequence, the specific relevance of categorial analyses is too often neglected.

The three levels of ontology are different but not separate. In many respects they affect each other. Descriptive findings may bear on formal categories; formalized outcomes may bear on their formal equivalents, etc. To set out the differences and the connections between the various ontological facets precisely is a most delicate but significant task (Poli, 2003).

1.3 Ontology_t

Ontological engineering, i.e., ontology from the perspective of computer science, has issues comparable to that of philosophical ontology, but reflected technologically in the attempt to develop ontologies as software usable models. So ontology from the perspective of computer science is both a computer science and a computational or software engineering problem. On the one hand, “ontological engineering”⁵ historically had its origins as an engineering problem, as an attempt to create software usable models of “the ways things are, with the things that are” to endow software with human level representations of “conceptualizations” or semantics. On the other hand, there are efforts that intend to make an “ontological science”, as for example, that of the National Center for Ontological Research (NCOR) (Obrst, Hughes and Ray, 2006).⁶ Such an effort would include strong evaluation criteria and possibly ontology certification.

Although having antecedents in the late 1980s, as formal ontology in philosophy and formal semantics in linguistics began to impact computer science and especially artificial intelligence, ontological engineering as a discipline can be marked as originating approximately in 1991, with Neches et al. (1991) reporting on the United States Defense Advanced Research Projects Agency’s (DARPA) Knowledge Sharing Initiative, and Gruber (1991), followed soon after by work by Gruber (1993), Guarino (1994), and Guarino and Poli (1995).

1.3.1 Ontology_t Definitions

The first proposed definition of ontology in computer science was that of Gruber (1993)⁷: “an ontology is an explicit specification of a conceptualization”, which

⁵The first occasion of use of the term “ontological engineering” is apocryphal: perhaps it occurred as part of the Cyc project (Guha and Lenat, 1990).

⁶National Center for Ontological Research (NCOR): <http://ncor.buffalo.edu/>.

⁷Anecdotally, the term “ontology” had been used in computer science and artificial intelligence since the late 1980s. One of the authors of this chapter described the use of ontologies and rules in Obrst (1989).

was intended to contrast with the usual definition of ontology in philosophy, i.e., to emphasize that what was being talked about was ontology_t in our terminology: ontology as a computational engineering product. The notion of “conceptualization” was defined in Genesereth and Nilsson (1987) to be “the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them” (Gruber, 1993) and presumably the “area of interest”, now typically called “domain”, is a portion of the world.

Guarino and Giaretta (1995) took up the challenge to clarify what was meant by this and other emerging definitions of ontology_t. In Guarino’s and Giaretta’s analysis, there were a number of ways to characterize ontology (quoted from Guarino and Giaretta, 1995, p. 25):

1. Ontology as a philosophical discipline
2. Ontology as an informal conceptual system
3. Ontology as a formal semantic account
4. Ontology as a specification of a conceptualization
5. Ontology as a representation of a conceptual system via a logical theory
 - 5.1 characterized by specific formal properties
 - 5.2 characterized only by its specific purposes
6. Ontology as the vocabulary used by a logical theory
7. Ontology as a (meta-level) specification of a logical theory.

By way of a summary: “ontology: (sense 1) a logical theory which gives an explicit, partial account of a conceptualization; (sense 2) synonym of conceptualization.” (Guarino and Giaretta, 1995, p. 32)

Note that characterization (4) invokes Gruber’s definition. Part of Guarino’s and Giaretta’s explication involves analyzing Gruber’s [derived from Genesereth and Nilsson’s (1987)] notion of a conceptualization as being extensional. Instead, Guarino and Giaretta (1995) argue that it should be an intensional notion. Rather than Genesereth and Nilsson’s (1987) view of conceptualization as “a set of extensional relations describing a particular state of affairs,” in Guarino’s and Giaretta’s view, it “is an intensional one, namely something like a conceptual grid which we superimpose on various possible states of affairs.” (Guarino and Giaretta, 1995) The definition that Guarino and Giaretta end up with is that an ontology is an ontological theory, and as such that it “differs from an arbitrary logical theory (or knowledge base) by its semantics, since all its axioms must be true in every possible world of the underlying conceptualization.”

1.3.2 Ontology_t and Epistemology

A further issue about ontology and epistemology should be brought out now, as it relates to ontology_t. We have mentioned that epistemology deals with how

knowledge is known. How do my perception and understanding, my beliefs, constrain my arrival at real knowledge or assumed belief, i.e., evidence, knowledge hypotheses prior to their becoming theorems about knowledge (and there should be a clear path from hypothesis to theorem to true theorem, but often there is not). So if an ontology is a theory about the world, epistemology addresses the ways of acquiring enough knowledge (and the nature of that) so that one can eventually frame a theory. In *ontology_t*, the engineering artifact of the ontology model (a theory) will require epistemological linkage to data. That data can be inaccurate, contain uncertainties, and lead to partially duplicate but inconsistent instances of ontology classes. Epistemology thus is employed in the use and qualification of data and as stored in databases or tagged or indexed in documents.

If ontology states that human beings have exactly one birth date, the data about a specific person is epistemological: in a given set of databases the person instance named John Smith (we assume we can uniquely characterize this instance) may have two or more attributed birth-dates, not one of which are known to be true. Epistemological concerns distort and push off needed ontological distinctions. Evidence, belief, and actual adjudication of true data is epistemological. What the real objects, relations, and rules are of reality are ontological. Without ontology, there is no firm basis for epistemology. Analysts of information often believe that all is hypothesis and argumentation. They really don't understand the ontological part, i.e., that their knowledge is really based on firm stuff: a human being only has one birth date and one death date, though the evidence for that is multivarious, uncertain, and needs to be hypothesized about like the empirical, epistemological notion it is.

In fact, much of so-called "dynamic knowledge" is not ontological in nature (ontological is relatively static knowledge), but epistemological. What is an instance that can be described by the ontology? How do I acquire and adjudicate knowledge/evidence that will enable me to place what I know into the ontological theory? Instances and their actual properties and property values at any given time are dynamic and ephemeral (this particular event of speaking, *speaking_event_10034560067800043*, just occurred; however the *speaking_event* ontology class has not changed).

1.3.3 *Ontology_t as Theory with Philosophical Stances*

Ontology_t often considers an ontology to be a logical theory about some portion of the world.⁸ Philosophical stance towards theories is therefore quite important, because a given ontological engineer will typically imbue his *ontology_t* engineering model with constructs aligned with his or her philosophical stance, e.g. as to the preferred theory of universals (nominalism, conceptualism or realism).

⁸See for example, the discussion of what an ontology is on the Ontolog Forum site: <http://ontolog.cim3.net/cgi-bin/wiki.pl?>, i.e., Obrst (2006).

1.4 Interplay Between Ontology_c and Ontology_t

The issues we discuss in this section involve the complex interplay between ontology_c and ontology_t. Two main points are discussed: (1) the proper way of developing formalized ontologies; (2) an illustration of one case in which the interplay between philosophy and computer science can be explicitly seen. We discuss the problem of the “natural” boundaries of a domain ontology and how different types of domain ontologies should be distinguished.⁹

We take both Guarino’s and Giaretta’s position (Guarino and Giaretta, 1995; ontology as interpreted formal system, i.e., a logical theory) and Gruber’s position (Gruber, 1993; ontology as specification of a conceptualization) as problematic. Concerning the former, we think focusing on ontology as interpretation only is insufficient. As reflected in Guarino and Giaretta (1995), in this view ontology is more focused on the interpretation (semantics) of a logical theory, i.e., has more of a conceptual-flavored and model-theoretic position ultimately. A consistent logical theory can be developed about nonsense, for example, with no intent to describe a portion of the real world, the task of philosophical ontology as we see it. Subsequent discussions by Guarino (e.g., Guarino, 1998a, 2002; Masolo et al., 2003) have pointed to a better reconciliation between logical theory and realist-based formal ontology, which is more closely aligned with our view, as discussed next.

Against both of these views, however, we would rather say that ontology starts to be something relevant only when specifically ontological axioms are added to some formal basis (say, FOL). The definition of new concepts without the introduction of new axioms has limited value. In this regard, we consider as exemplar the General Formal Ontology (GFO) (Herre et al., 2006).¹⁰ But we also admire other upper or foundational ontology efforts which have sought to axiomatize their distinctions, including, Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE),¹¹ Basic Formal Ontology (BFO),¹² Object-Centered High-level Reference Ontology (OCHRE), Suggested Upper Merged Ontology (SUMO),¹³ Upper Cyc,¹⁴ etc. Recently there was an effort to reconcile or at least map among many of these upper or foundational ontologies by the Ontolog Forum (Obrst et al., 2006).¹⁵

⁹We do not discuss ontological layers here in any detail. The interested reader instead is pointed toward the chapters on the Categorical Stance and on Ontological Architectures in this volume.

¹⁰General Formal Ontology (GFO): <http://www.onto-med.de/en/theories/gfo/index.html>. See also Herre’s chapter in this volume.

¹¹For DOLCE and OCHRE, see Masolo et al. (2003) and the site: <http://www.loa-cnr.it/DOLCE.html>.

¹²Basic Formal Ontology (BFO): <http://www.ifomis.uni-saarland.de/bfo>.

¹³Suggested Upper Merged Ontology (SUMO): <http://www.ontologyportal.org/>.

¹⁴Upper Cyc: <http://www.cyc.com/cycdoc/vocab/vocab-toc.html>.

¹⁵Ontolog Forum: <http://ontolog.cim3.net/cgi-bin/wiki.pl?>.

1.4.1 Developing Formalized Ontologies

Concerning the second issue, the interplay of ontology_c and ontology_t, we provide an example that illustrates some domain ontology distinctions. We initially assume that the basic distinction between domain ontologies (DO) and upper ontologies (UO) are given. We further assume that a boundary has been established between a selected UO and the DOs which are or could be subsumed under it. Consequently, it should be clear which concepts pertain to the UO and which pertain to its DOs. Typically, highly general concepts like “process”, “part”, and “boundary” are likely to be included in a UO, while concepts like “gene”, “cell” and “membrane” are likely to be included in a domain ontology for, say, biology. Note that we do suppose that a domain ontology for biology may be considered a domain-specific UO, since the constructs of the domain ontology may correctly have to be made general enough to encompass prospectively an entire science. Considering biology as a domain with respect to a true UO, then in turn a biology domain ontology may be considered a domain-specific UO with respect to many complex sub-domains. These sub-domains can be considered domains in their own right (perhaps also incorporating other domain ontologies, say that of public administration for the case of public health), given the complexity of their subject matter, e.g., mammalian anatomy, neuropathology, genetic engineering, clinical medicine, public health, pharmacology, etc. We might call such a domain-specific UO a middle ontology (that spans multiple domains), a “superdomain” ontology, or simply a domain-specific UO.

Figure 1.1¹⁶ depicts the basic layers and the nomenclature we employ. By “utility ontology” in the above, we mean an ontology that represents commonly used

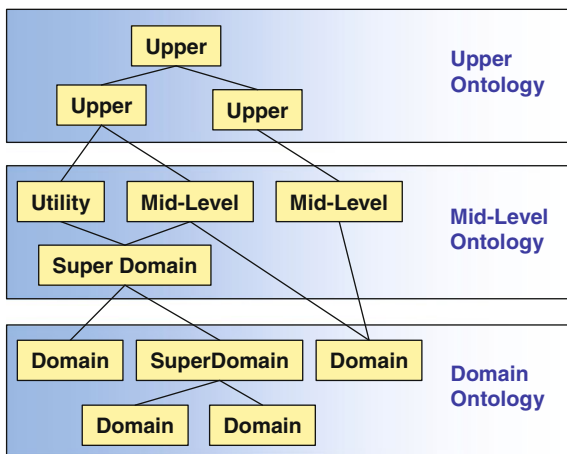


Fig. 1.1 Ontology layers

¹⁶From Fig. 9.1 in Chapter 8, Ontological Architecture; also see Semy, Pulvermacher and Obrst (2005, p. 8).

concepts, such as Time and Location. However, there is no crucial distinction between a “utility” and a “mid-level” ontology. We do note that in general, a mid-level ontology more concretely represents concepts that are defined more abstractly at the UO level.

To establish our ideas, the two following situations offer useful hints.

Case 1. At the beginning of the previous century the Polish-Russian philosopher of law Leon Petrazycki called attention to a basic theoretical requirement of theory development. We quote one relevant passage:

“Many theories, comprising no fallacy, are yet inadequate: one may form the concept of “a cigar weighing five ounces”, predicate about that class everything known about material things in general (about solid bodies in general, the chemical properties of the ingredients of these cigars, the influence of smoking them on health, and so on); these “theories” while perfectly correct are manifestly inadequate since what is predicated with respect to “cigars weighing five ounces” is also true of innumerable objects which do not belong to that class, such as cigars in general. A theory may be inadequate either (1) because the predicates are related to classes which are too narrow . . . or (2) because the predicate is related to a class which is too broad (such as various sociological theories which attribute “everything” to the influence of one factor which in fact plays a much more modest part)” (Petrazycki, 1955, p. 19).

We may well read “ontologies” where Petrazycki writes “theories”. In fact, one may well read “concepts”, since cognitive science has a comparable notion concerning “concepts”, i.e., that they be non-profligate in a similar manner; especially with respect to what is called the “theory–theory of concepts”, concepts as “mental theories” (Laurence and Margolis, 1999, p. 43), and with respect to profligacy: the potential concepts “the piece of paper I left on my desk last night”, “frog or lamp”, “31st century invention” (Laurence and Margolis, 1999, p. 36). Interestingly, it seems conceptual analysis is recapitulating ontology and semantics, since the former is also addressing categorization, analyticity, reference determination, and the notion of a “prototype” including the notion of “evidential” vs. “constitutive” properties (Laurence and Margolis, 1999, p. 33), which stumbles on the epistemology vs. ontology conundrum.

The point here is that an ontology (*viz.* a *domain* ontology) may then be inadequate if its boundaries are badly cut. But how should one know where to draw “natural” or “appropriate” boundaries? Some will say that many ontologies at the domain and middle levels correspond to scientific disciplines, i.e., that science and scientific theories apportion the areas of interest. This is partially true, but of course it dismisses intuitive or common-sense ontologies that humans may have, each even considered a logical theory about the world, because they are based on non-scientific generalizations. A theory of parenthood, for example, may not be scientific *yet*, i.e., not based on a combination of sociology, anthropology, biology, psychology, economics, political science, and everything else that might be scientifically known, but it may be a reasonable approximation of reality for the short or mid term, since it’s very doubtful those combination of scientific theories will be reconciled any-time soon. This point argues for the inclusion of commonsense theories in lieu of established scientific theories, the latter which may not ever be forthcoming.

Case 2. It is well known that for decades classification theory has labored under an unresolved split between two different methodologies. This split is particularly pronounced in the case of frameworks elaborated by librarians, where it takes the form of the difference between enumerative or taxonomical (Dewey Decimal Classification (DDC), Library of Congress Classification (LCC), etc) classification and faceted classification, also called colon classification, originating from the analysis of Ranganathan (1962). Since faceted classifications are now being proposed for Web construction and other computer-based applications as a more effective way to organize information, a proper understanding of its nature is becoming increasingly relevant. Unfortunately, what is not clear is whether any criteria is available for deciding whether an enumerative or a faceted style of classification should be adopted.

Where lies the *natural* boundary of an ontology_t? The question is both difficult and subtle. May it not be that the boundaries of an ontology may depend on subjective intentions? Just as semantics-pragmatics in linguistics and philosophy of language represents a spectrum, with the latter pole being focused on “semantics in context, with respect to a given use and intent”, there are subjective intentional issues here. However, the problem of subjective reasons or needs (“I am developing this ontology for this and that reason, based on these use cases”) subtly misses the point. Subjective motivations are always there, and they may more or less severely constrain the ontology to be build. We think that even moderate subjectivism is problematic. So it is for this reason that we don’t consider an ontology_t as a standard or an agreement: an ontology is not the result of a standards-based concensus of opinion about a portion of the world, because, in general, the effort and thus the result will devolve to the lowest common denominator, and generally end up worthless – because it is inconsistent, has uneven and wrong levels of granularity, and doesn’t capture real semantic variances that are crucial for adoption by members of a community. Users of ontology_t cannot be the developers of ontology_t, for much the same reason as users should not develop their own databases: users intuitively know their own semantics, but typically cannot express the ontological and semantic distinctions important to them, nor therefore model them – even though the real world referents are common to everyone.

This avoidance of subjectivism is notwithstanding the established or preferred methodology for developing a specific ontology_t, i.e., that one must focus on the use cases, anticipated scenarios that instantiate those, and therefore the software modeling requirements and “competency questions” (the queries you want answered, i.e., theorems with instantiations which make them true, or the queries you would like to have answered if it were possible for this new ontology-based system to provide you with such), as Fox and Gruninger (1994) and Uschold and Gruninger (1996) clarify. So ontology_t’s methodology is to proceed both bottom-up and top-down, i.e., analyze the data sources with respect to their semantics which will have to be captured by the resulting domain ontology and the questions end users (domain experts) would like to ask if they could (and can’t typically ask using database and mainstream computing technology). Concerning the latter, typically end users can’t formulate these kinds of questions because their imagination

is constrained by their current systems. It takes patient knowledge elicitation and knowledge of comparable kinds of value by the working ontology engineer to eke out this kind of knowledge question.

The *focus* of ontological analysis is not centered on the subjective intentions motivating the constructions of the ontology but on the item to be modelled. The reasons for which one is modeling this or that item (or class thereof) may and do interfere, even dramatically, but it is the item itself that is relevant. It is the thing in the world that the ontology is grounded on.

The problem is the old philosophical problem of the connections between epistemology and ontology_c, as we mentioned earlier. The problem has been made obscure by the attitude widely prevalent in recent mainstream philosophy according to which epistemology prevails over ontology.

Ontologies_t could be more or less detailed; their existence may even – at least in some cases – modify the functioning of the modelled system. However, the main question is: does the *existence* of the item/system under observation depend on the ontology? When the answer is negative – as it is for the overwhelming majority of cases – we have a basis for severing the ontological core from all the rest.

The first criterion is then to look for what exists. A number of relatively easy qualifications should now be taken into consideration. The easiest one is to consider only directly observable items, i.e. actually existent items, but also items that existed in the past. They are no more directly observable but we could observe them were we living at their time (This is the pragmatist criterion firstly devised by Peirce. The case where one can *now* observe the traces left by no more existent items is trivially unproblematic). By adopting the same criterion, one may eventually include also items possibly existing in the future.

More demanding is the question about what is said to exist, i.e., the primary interest of ontology_c. For example, we may say that there are material things, plants and animals, as well as the products of the talents and activities of animals and humans in the world. This first prosaic list already indicates that the world comprises not only things, animate or inanimate, but also activities and processes and the products that derive from them. For human-developed products, for example, functional properties are significant (a claw hammer is meant to pound and remove nails; its head and handle are therefore of length and material composition that is appropriately leveragable for those operations). It is likewise difficult to deny that there are thoughts, sensations and decisions, and in fact the entire spectrum of mental activities. Similarly, one is compelled to admit that there are laws, languages, and factories.

We can set about organizing this list of items by saying that there are material items, psychological items and social items (Poli, 2001a), as displayed in Fig. 1.2 below, which depicts dependence of these categories.¹⁷ In turn, each of them presents a vast array of subtypes (material items include physical, chemical, and

¹⁷Poli's Ontology: The Categorial Stance (TAO-1) discusses these issues in more detail.

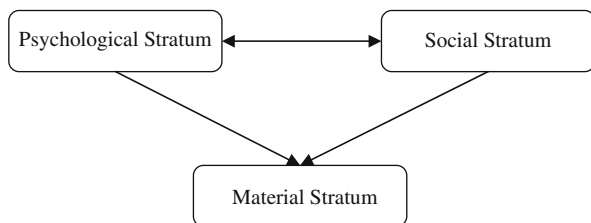


Fig. 1.2 Ontological strata

biological items, psychological items include representation and emotions, social items include laws, languages and many other types of pertinent items).

This section started by asking the natural boundaries of an ontology, how do we determine what an ontology includes or does not include? In trying to provide an answer we found ourselves involved in classical philosophical problems, which is not at all surprising.

1.4.2 Ontology, Science, and Levels of Reality

Returning to our main question, any possibly correct answer concerning what an ontology should include will have to start articulating its proposal with respect to some existing item (or type of). Subsequent steps may follow a variety of different paths. However, for most cases one route seems particularly prominent: that adopted by science. For apparently good reasons, science has been developing in different branches (including physics, economy, biology and cognitive science), the idea being that there are classes of items that “go together”, constituting at least a description and possibly an explanation over some portion of reality. In this regard, ontology may follow the same route successfully traversed by science. However different they are, ontology and science are allies. This view intends to convey that between ontology and science there is a mutual exchange, from science to ontology and from ontology to science. That ontology may have something to offer science can be seen from the idea of an ontologically reconstructed and clarified science.

The suggestion is therefore to start from well established scientific partitions. Even if something more will later be required, this initial step will nevertheless help in avoiding two opposed risks. A truly atomistic vision claims that atoms are the only authentically existing items, and that all the other items we recognise are ephemeral. On the other hand, the followers of a boldly holistic vision will claim that the only autonomously existing item is the universe as a whole. Neither of these visions suits ontology. By relying on the multiplicities of sciences one automatically advocates a molar strategy: there are many different items, of different types, requiring different categorial frameworks.

So far so good. The point we arrive at, however, represents both a safe result and one of maximal difficulty. As a matter of fact, so far modern science has relied on

an essentially analytic strategy. Different sciences have been developed in order to efficaciously segment the whole of reality into classes of more or less uniformly connected phenomena. The guiding idea has been that phenomena occurring within each class are more causally homogeneous than phenomena pertaining to other classes, so that the task of explaining their behavior should be more easily accomplished. This divide and conquer strategy has proved immensely successful, at least for some regions of reality. Other regions have proved more refractory, for a number of serious reasons. The first is that different regions may require different types of causation, some of which are still unknown, or only partially known (Poli, 2007). A second reason is that for some regions of reality the analytic strategy of breaking items into pieces does not work properly. A third and somewhat connected reason is the lack of a synthetic methodology.

The complexity of reality requires the analytic strategy of segmentation into *categorially* homogeneous regions. This first move is not questioned. However, some regions contain only items that can be further analytically segmented into pieces. These items are entirely governed by their parts (from below, so to speak). Other regions contain items following different patterns: they depend on both their parts and the whole that results from them. Our understanding of these more complex items is still deficient. Recent theories about *granular partitions* (Bittner and Smith, 2001, 2003; Bittner et al, 2007; also see Rogers and Rector, 2000) attempt to remedy this situation.¹⁸ Even so, unfortunately, this is not the end of the story. Something more is further required: sooner or later the products arising from the segmentation into categorially homogeneous regions should be synthesized. For we all live in *one* world. This second synthetic move has proved much more troublesome than the original analytic move.

A properly developed synthetic strategy still awaits us. However, the theories of levels of reality may represent a helpful step toward the elaboration of a fully developed synthetic strategy.¹⁹ Each layer of reality requires (1) specific kinds of items, (2) appropriate categories, and (3) links to its bearing (i.e. based on building-above

¹⁸Bittner and Smith's (2003) framework tries to uphold the strengths of set theory and mereology for modeling parts and wholes but avoid their respective weaknesses by building on the distinction between *bona fide* (objects which exist independently of human partitioning efforts and *fiat* objects (objects which exist only because of human partitioning efforts) (Smith, 2001). As such, their theory of granular partitions begins to impinge on the distinction too between the semantic notions of intension and extension – because on one view, two intensional descriptions (“the morning star”, “the evening star”) can be seen as human partitions, even though both extensionally refer to the same object, Venus. In their view, “partition is a complex of cells in its projective relation to the world” (Bittner and Smith, 2003, p. 10), and so a triple is established: a granular partition, reality, and the set of “projections” or mappings to and from the items of the partition and reality. Whether this is ontology or ontology intermixed with epistemology remains to be clarified.

¹⁹Note that we use “level” to refer in general to the levels of reality, restricting the term “layer” to over-forming relationships, and the term “stratum” to building-above relationships. The interested reader is directed to Poli, “Ontology. The Categorical Stance” (TAO-1) for a fuller exposition of this topic.

relations (*Überbauung*), and conditioning (i.e. based on over-forming relations, or *Überformung*)²⁰ layers as described in [Chapter 1](#), TAO_1.

These are precisely the lacking elements needed to answer the question above asked on the natural boundaries of an ontology: the boundaries of a domain ontology are the top domain categories needed for defining the domain items, plus eventual bearing and conditioning links.

1.4.3 Example: An Ontology of Biology

Using our suggested methodology, we now describe the domain ontology of biology.

Many ontologies_t have been recently developed in the field of biology. Most of them are found at the OBO website.²¹ Bio-ontologies offer a nice case for discussing ontological integration. By looking at the ontologies collected by the OBO initiatives one may wonder whether (and how) they could be coordinated and eventually integrated. The problem we would like to address is whether a methodology – here understood as a set of instructions or guidelines – could be devised for developing easy-to-integrate ontologies. Generally speaking, this will comprise minimally three cases: (1) *vertical integration*, i.e., specific ontologies integrable within more general ontologies; e.g. anatomy within biology, (2) *horizontal integration*, i.e., integration among ontologies modelling categorially disjoint phenomena, e.g., business and legal ontologies, and (3) *cross-domain ontologies*, where a number of ontologies pertaining to different levels of reality should be both joined and pruned, e.g., medicine, which may require chemical, biological, psychological, economic, legal and religious information, among others. Some of the mentioned ontologies may further have to be pruned in order to include only information relevant to human pathologies.

The domain top level of our proposed biology ontology will be based on the following three concepts (Hohendorf et al., 2008):

- *Biological entity (BE)*.
- *Living entity (LE)*
- *Organism (OR)*.

Any biological item is a *biological entity*. The concept of BE refers to anything organic: DNA, mRNA, the nucleus of a cell, the membrane of a cell, its organelles, urine are bioentities. The main function of the concept of BE is to delimit the field. This will prove especially relevant when different ontologies_t are merged together. If all the merged ontologies_t define their most general domain concepts, their management will prove much easier (and safer).

²⁰Over-forming relations (*Überformung*) and building-above relations (*Überbauung*) are from Hartmann (1952).

²¹Open Biomedical Ontologies (OBO) Foundry. <http://obofoundry.org>.

Living entities are a specific class of BEs. Two features distinguish LEs from BEs (1) all LEs are systems, while not all BEs are; and (2) LEs are *metabolic* systems. This means that LEs are entities that can survive if appropriate nutrients are provided. Cells, tissues and organs are cases in point.

Lastly come organisms. These are *autopoietic* LEs, i.e. LEs able to produce BEs of which they are composed. Note that a different criterion could be also used: ORs are autonomous LEs (i.e., entities able to survive without a wider encompassing entity). However, we'll use autonomy shortly and will then include it anyway. We have two main types of ORs: unicellular and multicellular ones. The cell that is a unicellular organism is a BE, a LE and an OR. The multicellular organism as a whole is a BE, a LE and an OR. The difference lies in the cell composing the multicellular organism: differently from the single cell that is a unicellular organism, the cells of multicellular organisms are living entities but *are not organisms*. According to the proposed definitions, a liver cell, then, is a living entity but not an organism.

Two more issues remain to be addressed. The first issue concerns the special status of the cell, which sometimes is taken as a full-fledged organism and sometimes not. To mark the difference it is sufficient to consider that all the other living entities apart from cells can *never* be taken as organisms. Cells, therefore, have a special status, that should be marked in some way. We will mark this fact by using the same relation for both the organism-organism case and the cell (as LE) and the organism case. Table 1.1 below shows the relevant cases for binary relations.

Finally, one may notice that if one subtracts LEs from the field of BEs what remains can be taken as coincident with the field of organic chemistry. This is entirely correct and shows the link between a bio-ontology and the ontology/ies characterizing its underlining levels of reality. However, there are differences that shouldn't be forgotten. The most obvious one is that the subtraction of living entities modifies the situation in such a way that a number of question become unanswerable. Even if urine is a purely chemical substance (well, a mixture of), how could one explains its presence without taking organisms into consideration? Organic chemistry and biology present areas of overlapping, and that provides the link between them. However, the questions that are asked from within organic chemistry and the questions that are asked from within biology are different, and this shows that they are different. The reason because they are different lies in the entities grounding their specific levels of reality: molecule for chemistry, (cells and) organisms for biology.

All of this is obviously a first step only. Many other kinds of biological information need to find their proper place. Here is where the second case above mentioned enters the scene. Needless to say, the minimal structure based on the three categories of biological entity, living entity and organism should be widely enlarged if

Table 1.1 Relations between organisms and cells

Relations between unicellular organism A and unicellular organism B.
Relations between unicellular organism A and multicellular organism B (or vice versa).
Relations between cell A (as a LE and not as a OR) and multicellular organism B (or vice versa).

a reasonably articulated model of biology is going to be realized. Once the basic ontological structure of a domain has been established – that is to say, once the levels of reality of the domain have been fixed – the subsequent step is to devising their dimensions of analysis. Here is where faceted analysis can best play its role. Maintaining fixed our reference domain of biology, two series of facets follow. The first series is centered on the governing concept of organism as an individual whole and lists the “viewpoints” from which organisms so taken can be seen. One can then consider at least the following three cases:

- *Classification*
- *Structure*
- *Function*

Classification will model the unquestionably well known biological taxonomies, starting from the distinction between prokaryotes and eucaryotes and then substructuring the former into archea and probacteria and the latter into protista, fungi, plantae, and animalia.

Structure applies part-whole analysis to cells and organisms, for both parts, organ and tissues. Traditionally this type of analysis is called cytology for the cell and anatomy for multicellular organisms. A properly organized biological structure should comprise non only default cases but also variations and anomalies.

Function, finally, corresponds to what is traditionally called physiology and model the working of the part descriptively listed by the previous facet of the organism’ structure. In the case of the facet of functions, (serious) variations are usually called pathologies (to be further distinct between intra-systemic and inter-systemic pathologies).

The second series of facets list all the other viewpoints, those not focused on the organism as a whole. These may comprise for instance *genetics* (focus on the genes), *ethology* (focus on some population of organisms), *ecology* (focus on an entire ecosystem). But, again, this is not the entire story. A substantial number of other facets can and should be developed, concerning for instance the *growth* and *development* of organisms, or their *reproduction*, or their *alimentation*. For each of these facets, appropriate ontologies can be developed.

It is time to sum up what has been described in the sections above. We will now try to extract a general scheme from the various topics so far seen.

We propose to distinguish four different types of domain ontologies:

1. *Domain ontologies* in the proper sense (e.g. Biological ontology)
2. *Sub-domain or facet ontologies* (e.g. Gene ontology)
3. *Cross-domain ontologies* (e.g. Medical ontology)
4. *Micro-domain ontologies* (e.g. an ontology of edible substances)

In order to maximize the likelihood of ontology mapping, merging, i.e., integration, we advance two different claims: (1) domain ontologies (of any of the above-distinguished types) should use a top level ontology as their best change of

being grounded in a robust framework; (2) furthermore domain ontologies should contain their own top level (domain-top-level); we will further claim that each of the four domain ontologies above distinguished needs a different domain-top-level (this partially explains why distinguishing domain ontologies into specific types is relevant).

In order to distinguish the different cases, the first needed step is to find a criterion for distinguishing domain ontologies in the proper sense from the remaining cases. As a preliminary working definition we propose to define a proper domain ontology as a (1) categorially closed, (2) maximal partition of reality. Therefore, not every partition whatsoever is a domain (in the proper sense).

Our second partition is between domain ontologies (in the proper sense) and their facets. Consider the above described case of a biological ontology. Our claim is that bio-ontology as a whole is a domain ontology while taxonomy, anatomy, physiology, genetics, ecology etc. are some of its facets (and therefore should be properly classified as facet ontologies and not as domain ontologies).

We have claimed that a domain ontology is a categorially autonomous level of reality. "Categorially autonomous" means that even if its phenomena/entities may be existentially dependent from lower and/or side level entities, they nevertheless require a categorial framework different from the one used for understanding the entities of the existentially supporting levels (i.e., they are categorially autonomous).

For instance, biological entities (organisms) require chemical entities (molecules) which in their turn require physical entities (atoms) as their "matter". However, the frameworks needed for understanding biology, chemistry and physics are different (otherwise there will be only one single science). A somewhat more complex case is provided by side-level domains: the connection linking economy and law is different from the hierarchical one linking say chemistry and biology. The former are domains based on specific phenomena and each requires its specific categorial framework. However, the one supports existentially the other; none of them can exist without the other. Furthermore, both require underlining existential support for both agents and their biological environment.

The above makes clear that one should distinguish existential dependence from categorial autonomy. The latter literally means that the field under analysis presents phenomena requiring new categories.

Any ontology (domain or not) presents a number of different phenomena and usually requires a vast array of categories. Here we claim that all domain ontologies present one (or very few) basic (types of) entities. Sometimes they are difficult to find. Occasionally, science itself has needed quite a while before discovering the constitutive (dominant) entity/entities of the field. The list of domain-constituting entities is an enormously powerful tool for ontology development.

Finally, the top level of a domain ontology should comprise the following information (in parentheses is the information for bio-ontology):

1. The most general domain categories (BE, LE, OR)
2. Link from the domain's levels of reality (Organism) to the Theory of Levels module of the general top level ontology

3. The namespaces for the domain's "facets" and their position in the overall structure of the domain ontology

The top level of a sub-domain ontology will include only (1) and (3), but may lack (2). (3) may be needed in case the sub-domain ontology is further segmented into sub-sub-domain ontologies.

The last case is the case of cross-domain ontologies, as e.g. medicine. The minimal indication we can here provide is that the top level of a cross-domain ontology should include the top levels of all its relevant domains, plus appropriate pruning of the domain's internal organization. The most obvious case of pruning is limiting biological information to those fragments of biology that are relevant for *homo sapiens*.

1.5 Looking Toward the Future

In this chapter we have demonstrated the interaction between ontology_c and ontology_t, elaborating the definitions of each and describing issues pertinent to both. In addition, we have illustrated this collaboration with an extended example, developing the foundations for an ontology of biology.

Both ontology_c and ontology_t are dependent on each other: (1) ontology_t depends on ontology_c for clarification of the issues involved in describing ontologies about the real world, and (2) ontology_c depends on ontology_t for representing the ontologies about the real world that can be developed into engineering products and be used by software, enabling machines to more closely interact at the human level of conceptualization.

We see increasing interaction between these two aspects of ontology, ontology_c and ontology_t, in the future. Once the more basic issues are elaborated on and scientific consensus established (as we have broached in this chapter), issues such as ontology modularity, mapping, context-determination and representation, vagueness and uncertainty, and ontology lifecycle development will become predominant.

What will become more important are issues ontology_c and ontology_t can work on together and evolve solutions for. Some of these are among the following.

1.5.1 Better Ordering Relations for Ontologies

One issue is the nature of the order relations for ontologies. These can be characterized in many ways: from the perspective of mathematics and computer science set-theoretically, i.e., as partially ordered sets, lattices (semi-lattices; Davey and Priestley, 1991), and including structures used by formal concept analysis (Ganter and Willey, 1996), etc.; or category-theoretically (MacLane, 1971; Lambek and Scott, 1986; Pierce, 1991; Asperti and Longo, 1991; Crole, 1994; Barwise and Seligman, 1997), etc. In ontology_c, these order relations may include those above

but extended with notions of mereology and mereotopology (Varzi, 1998), granular partitions (Bitner and Smith, 2003), strata and levels of reality (Poli, 1998, 2001a).

A related issue concerning order is a prospective reconciliation of the notion of subsumption (set-theoretic order) heavily used by informal taxonomies, description logics, thesauri, object-oriented conceptual models, and the logical theories of ontology engineering. Typically this subsumption relation is called *subclass_of* or *isa*, i.e., the special transitive relation of a child class (non-terminal category) to its parent class, with the implicit assumption being made that there is some necessary property that distinguishes a child class from its parent or siblings classes.²² The subclass/isa relation forms a taxonomic backbone for ontologies_t. Ontology_c has influenced methodologies and tools to assist in the improved development of the taxonomies at the core of ontologies, e.g., the use of meta-properties in OntoClean (Guarino and Welty, 2002).

1.5.2 Elaboration of the Distinctions Among Ontology Levels

We envision increased collaboration in the future between ontology_c and ontology_t on some of the issues we began to elaborate in Section 1.4 focusing on ontology architecture, e.g., the distinctions between upper ontologies and domain ontologies, including levels of reality, and distinct theories of causality (Poli, 2007). These issues also impact the part/whole distinctions made by meronymy, mereology, and topology (Simons, 1988); Varzi and Pianesi, 1996a, b; Varzi, 1998). Discussion of granularity brings up issues related to zooming in and out and reasoning at different levels of reality, approximation, and vagueness (see Williamson, 1998; Keefe and Smith, 1999; Obrst and Mani, 2000; Varzi, 2000; Bittner and Smith, 2001).

Upper ontology issues of ontology_c have already come to the forefront in developing ontology_t products, including time/space distinctions and perspectives: 3D vs. 4D, endurantism vs. perdurantism (see Chapter 14 by Herre, this volume), SNAP/SPAN notions of Grenon (2003), Grenon and Smith (2003). In many cases, ontology engineers working in ontology_t are fiercely supportive of one perspective or another; others would like to maintain the multiple perspectives, picking and choosing which upper theories to use for different domain ontologies or different domain applications. To do so, bridge axiomatization are necessary, e.g., bridging axioms relating 3D to 4D would provide the best of both worlds. However, to date no one has actually created these bridge axioms (Hamlyn, 1984; Sider 2001; Loux, 2002; Obrst et al., 2006).

²²There is also the *instance_of* relation that is the relation between a lowest-level class (non-terminal) or classes (in the case of multiple parents) and the instance (terminal, an individual or particular) which instantiates the properties of that class or classes. In general, classes are universals and instances are particulars.

1.5.3 *Ontology Modularity, Mapping, and Formalization of Context*

Ontologies_t are often called *logical domain theories* to emphasize that they are logical theories about specific domains of the real world. In the case of upper ontologies, these are *logical foundational (upper) theories*, to signify that they are about notions that traditionally originate from philosophical ontology, ontology_c, but are no less logical theories, i.e., syntactic theories (with licensed semantic models) expressed in a logic and similar to scientific theories except also often extended to common-sense reality (where no scientific theory yet exists or perhaps ever will), and thus a common-sense theory.

Viewed as a collection of interlinked logical theories, ontology_t is concerned with establishing the nature of the relations among these interlinked logical theories, i.e., the nature of the links. Mathematically and computationally, these links are important because they characterize notions of modularity among and within ontologies. If micro-theories can be established (and represented as engineering products) and linked, then these micro-theories can be seen as constituting a theory. Together, many theories can constitute larger theories about reality and on which automated reasoners and other applications in information technology can operate.

It's important that these links among micro-theories and theories are defined logically, so that the "correct entailments" flow across those links. Automated reasoners that reason on vastly many theories require logical notions of modularity, to ensure sound and consistent reasoning. But the definition of what constitutes an ontological "module", i.e., a micro-theory or theory, is not yet agreed on. There are many candidates: one notion of modularity is that of "little theories" in mathematics (Farmer et al., 1992), e.g., how the theory of monoids are related to the theories of groups. There is the microtheory of Cyc (Blair et al., 1992). There is the modularity of category theory (Kent, 2004), which focuses on categories and systems of categories and morphisms among them. There is the so-called "lattice of theories" approach in which distinct ontologies are related by logical relations (or their interpretations) (Sowa, 2000).

But are ontologies_t logical theories about the world or are they vocabularies/models about a community? Some in the information technology community view ontologies as collaborative agreements, more like common conceptual models or standards. This is not a general view, but it exists. Hence, communities of interest form to share information and do so by developing common vocabularies. Typically this is a bottom-up paradigm, wherein communities form to share at least a subset of their individual information, and require common vocabularies and models of those vocabularies. Frequently this paradigm views the process of eliciting these vocabularies and models as a standards activity, wherein consensus is established among a potentially large community.

Directly related to modularity is the the notion of mappings among ontologies, in which disparate ontologies are related. Mapping includes issues such as enforcing local consistency but tolerating global inconsistency (Wiederhold, 1994; Mitra

et al., 2000; Noy and Musen, 2000). In general, the mapping between two domain ontologies constitutes a third *integrative domain ontology* which must be able to express everything that is in the two source ontologies.

Finally, pertinent to modularity within and among ontologies is the notion of formalization of context: what does it mean to index assertions of ontologies, to state that certain propositions are true in a given context? Is this the same notion as that of possible worlds with accessibility relations among worlds, with some worlds being “farther away” than other worlds because of their greater respective inconsistencies? Do we need modal logic to express this? Contexts are sometimes called views or perspectives; is this related to the thesaural notion of facets – i.e., perspectives or distinguishable trees connected by cross-references? Context will be increasingly relevant to automated reasoning over ontologies and their knowledge bases. For semantic, computational, and logical approaches to context (see Lewis, 1980; McCarthy, 1987, 1990, 1993; Guha, 1991; McCarthy and Buvač, 1997; Blair et al., 1992; Akman and Surov, 1996, 1997; Giunchiglia and Ghidini, 1998; Menzel, 1999; Bouquet et al., 2003; Obrst et al., 1999a, b; Smith and Obrst, 1999; Obrst and Nichols, 2005; Fikes and Welty, 2006).

1.5.4 Representation vs. Reasoning

This chapter has been primarily focused on the interplay between ontology_c and ontology_t with respect to *representation*, i.e., the correct explication of ontology from the perspective of philosophy and a coherent rendering of that into an engineering model that information technology can utilize. *Representation* here is commonly considered the content and the logical language that the content is expressed in. However, reasoning over that representation is especially important for ontology_t, just as it is for ontology_c. For ontology_c, the reasoning is performed by human beings; for ontology_t, the reasoning is performed by machines. We observe the slogan “No reasoning without representation”, which means that automated reasoning, much like human reasoning using formal logical argumentation, is ultimately constrained by the representation that is reasoned over, i.e., the content and the logical language the content is expressed in. One cannot perform full predicate calculus reasoning over content expressed in the propositional calculus, for example.

So a final consideration for future interplay between ontology_c and ontology_t is that focused on the nature of automated reasoning, i.e., the preferred logics for the reasoning, the types of automated reasoning one can perform on those logics (deduction, induction, abduction), the semantics-preserving transformations from one logic or knowledge representation language to another, including perhaps that to special substructural logics (Restall, 2000) for specific kinds of reasoning, and finally, issues of particular relevance to ontology_t concerning *knowledge compilation* (Kautz and Selman, 1994; Cadoli and Donini, 1997; Darwiche and Marquis, 2002), i.e., how best to make an efficient runtime representation for the automated reasoning – which necessarily addresses issues in computational and descriptive

complexity (Graedal et al., 2007) from computer science, while at the same time addressing issues in approximation and vagueness on which ontology_c can offer significant insight.

1.5.5 Final Words

Ontology_t needs to be informed by ontology_c and its methods. Ontology_c will increasingly benefit from the sound and consistent software engineering products arising from ontology_t. Computer science, formal philosophy, and formal semantics have come together to birth the beasts called “ontology science” and “ontology engineering”. These are strange beasts of Earth but are classifiable and describable under the heavens. Tomorrow it may be that our machines are thereby enabled to in turn be born as stranger beasts, which may yet interact with us human beings as cousins on the well-founded and computationally represented ontological firmament of Earth.

Acknowledgments and Disclaimers

One author’s affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE’s concurrence with, or support for, the positions, opinions or viewpoints expressed by this author. We note that the views expressed in this paper are those of the authors alone and do not reflect the official policy or position of any other organization or individual.

References

- Akman, V. and S. Mehmet. 1997. The use of situation theory in context modeling. *Computational Intelligence* 13(3):427–438, August, 1997.
- Asperti, A. and G. Longo. 1991. *Categories, types and structures*. Cambridge, MA: MIT Press.
- Barwise, J., and J. Seligman. 1997. *Information flow: The logic of distributed systems*. Cambridge, UK: Cambridge University Press.
- Basic Formal Ontology (BFO). <http://www.ifomis.uni-saarland.de/bfo>
- Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. 2004. OWL Web Ontology Language Reference. W3C Recommendation 10 Feb 2004. <http://www.w3.org/TR/owl-ref/>
- Bittner, T., B. Smith, and M. Donnelly. 2007. The logic of systems of granular partitions. Manuscript. <http://ontology.buffalo.edu/smith/articles/BittnerSmithDonnelly.pdf>
- Bittner, T., and B. Smith. 2003. A theory of granular partitions. In *Foundations of geographic information science*, eds. M. Duckham, M.F. Goodchild, and M.F. Worboys, 117–151, London: Taylor and Francis Books.
- Bittner, T., and B. Smith. 2001. Granular partitions and vagueness. In *Formal ontology and information systems*, eds. C. Welty, and B. Smith, 309–321, New York, NY: ACM Press.
- Blair, P., R.V. Guha, and W. Pratt. 1992. *Microtheories: An ontological engineer’s guide*. Technical Report Cyc-050-92, 5 Mar 1992, Cycorps, Austin, TX. <http://www.cyc.com/tech-reports/cyc-050-92/cyc-050-92.html>

- Bouquet, P., F. Giunchiglia, F. Van Harmelen, L. Serafini, and H. Stuckenschmidt. 2003. *C-OWL: Contextualizing ontologies*. In 2nd International Semantic Web Conference (ISWC 2003), eds. D. Fensel, K.P. Sycara, and J. Mylopoulos, 164–179, Sanibel Island, FL, 20–23 Oct 2003.
- Cadoli, M., and F.M. Donini. 1997. A survey on knowledge compilation. AI communications. *The European Journal for Artificial Intelligence* 10:137–150.
- Crole, R.L. 1994. *Categories for types*. Cambridge: Cambridge University Press.
- Daconta, M., L. Obrst, K. Smith. 2003. *The semantic web: The future of XML, web services, and knowledge management*. New York, NY: Wiley, June 2003.
- Darwiche, A., and P. Marquis. 2001. A knowledge compilation map. <http://www.cs.ucla.edu/~darwiche/d116.pdf>. An earlier version appeared as “A Perspective on Knowledge Compilation.” In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI’01), Seattle, WA, 175–182.
- Davey, B.A., and H.A. Priestley. 1991. *Introduction to lattices and order*. Cambridge, UK: Cambridge University Press.
- Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). <http://www.loa-cnr.it/DOLCE.html>
- Farmer, W.M., J.D. Guttman, and F.J. Thayer. 1992. Little theories. In *Automated deduction – CADE-11*, LNCS, vol. 607, ed. D. Kapur, 567–581. <http://imps.mcmaster.ca/doc/major-impapers.html>
- Fikes, R., and C. Welty. 2006. Interoperable knowledge representation for intelligence support (IKRIS). Advanced Research and Development Activity (ARDA)/Disruptive Technology Office (DTO). Final briefing, Nov 2006.
- Fox, M., and M. Gruninger. 1994. *Ontologies for enterprise integration*. In Cooperative Proceedings of the 2nd Conference on Cooperative Information Systems, Toronto, ON.
- Ganter, B., and R. Wille. 1996. *Formal concept analysis: Mathematical foundations*. Berlin, Heidelberg, New York: Springer.
- General Formal Ontology (GFO). <http://www.onto-med.de/en/theories/gfo/index.html>
- Genesereth, M.R., and Nilsson, N.J. 1987. *Logical foundations of artificial intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.
- Graedel, E., P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. 2007. *Finite model theory and its applications*. Heidelberg: Springer.
- Grenon, P. 2003. Spatio-temporality in basic formal ontology: SNAP and SPAN, upper level ontology, and framework of formalization (part I). Technical Report Series 05/2003, IFOMIS.
- Grenon, P., and B. Smith. 2003. Snap and span: Towards dynamic geospatial ontology. *Spatial Cognition and Computation* 4(1), forthcoming.
- Gruber, T. R. (1991). The role of common ontology in achieving sharable, Reusable knowledge bases. In Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, eds. J.A. Allen, R. Fikes, and E. Sandewall, 601–602, Cambridge, MA: Morgan Kaufmann.
- Gruber, T. 1993. A Translation approach to portable ontology specifications. *Knowledge Acquisition* 5:199–220.
- Guarino N. 1994. The Ontological Level. Invited Paper Presented at IV Wittgenstein Symposium, Kirchberg, Austria, 1993. In *Philosophy and the cognitive sciences*, eds. R. Casati, B. Smith, and G. White, Vienna: Hölder-Pichler-Tempsky.
- Guarino, N., ed. 1998. *Formal ontology and information systems introduction to formal ontology in information systems*. Proceedings of the First International Conference (FOIS’98), 6–8 June 199, Trento, Italy, 3–18. Amsterdam: IOS Press.
- Guarino, N., and Giaretta, P. 1995. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards very large knowledge bases: Knowledge building and knowledge sharing*, ed. N. Mars, 25–32, Amsterdam: IOS Press.
- Guarino, N., and R. Poli, eds. 1995. Formal ontology in information technology. Special issue of the *International Journal of Human-Computer Studies* 43(5/6). <http://www.ladseb.pd.cnr.it/infor/Ontology/IJHCS/IJHCS.html>

- Guarino, N., and C. Welty. 2002. Evaluating ontological decisions with OntoClean. *Communications of the ACM* 45(2):61–65. New York, NY: ACM Press. <http://portal.acm.org/citation.cfm?doid=503124.503150>
- Guha R.V. 1991. Contexts: A formalization and some applications. PhD Thesis, Stanford University. Also technical report STAN-CS-91-1399-Thesis, and MCC Technical Report Number ACT-CYC-423-91.
- Guha, R., and D. Lenat. 1990. Cyc: A mid-term report. Microelectronics Technology and Computer Corporation (MCC), Austin, TX. Technical Report ACT-CYC-134-90.
- Hamlyn, D.W. 1984. *Metaphysics*. Cambridge: Cambridge University Press.
- Hartmann, N. 1975. *The new ways of ontology*. Westport, Conn.: Greenwood Press.
- Herre, H., B. Heller, P. Burek, R. Hoehndorf, F. Loebe, and H. Michalek. 2006. General formal ontology (GFO), part I: Basic principles, Version 1.0. Deliverable No. 8 – July 2006. <http://www.onto-med.de/en/publications/scientific-reports/om-report-no8.pdf>
- Hohendorf, R., F. Loebe, R. Poli, H. Herre, and J. Kelso. 2008. GFO-Bio: A biological core ontology. *Applied Ontology* 3(4):219–227.
- Husserl, E. 2001. Logical investigations, vols. 1 and 2, Trans. J.N. Findlay with a new Preface by Michael Dummett, edited with a new introduction by Dermot Moran, Routledge, vols. 1 and 2.
- Kautz, H., B. Selman. 1994. An empirical evaluation of knowledge compilation. Proceedings of AAAI-94, Seattle, WA, July 1994.
- Kent, R. 2004. The IFF foundation for ontological knowledge organization. In *Knowledge Organization and Classification in International Information Retrieval*, eds. N.J. Williamson, and C. Beghtol, volume 37 of Cataloging and Classification Quarterly, 187–203. New York, NY: Haworth Press.
- Interoperable Knowledge Representation for Intelligence Support (IKRIS). <http://nrcc.mitre.org/NRRC/ikris.htm>
- ISO Common Logic. Common logic standard. <http://cl.tamu.edu/>
- Keefe, R., and P. Smith, eds. 1999. *Vagueness: A reader*. Cambridge, MA: MIT Press.
- Lewis, D. 1980. Index, context, and content. In *Philosophy and grammar*, eds. S. Kanger, and S. Ohman. Dordrecht: Reidel Publishing.
- Lambek, J., and P. Scott. 1986. *Introduction to higher order categorical logic*, volume 7 of Cambridge Studies in Advanced Mathematics. Cambridge: Cambridge University Press.
- Loux, M.J. 2002. *Metaphysics: A contemporary introduction*, 2nd edn. London and New York, NY: Routledge.
- Mac Lane, S. 1971. *Categories for the working mathematician*, volume 5 of Graduate Texts in Mathematics. Heidelberg: Springer.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. Wonderweb deliverable D18: Ontology library (Final). Technical report, Laboratory for Applied Ontology – ISTC-CNR, Trento.
- Margolis, E., and S. Laurence. 1999. *Concepts: Core readings*. Cambridge, MA and London: MIT Press.
- McCarthy, J. 1987. Generality in artificial intelligence, *Communications of the ACM* 30(12):1030–1035.
- McCarthy, J. 1990. *Formalizing common sense: Papers by John McCarthy*. Norwood, NJ: Ablex Publishing Corporation.
- McCarthy, J. 1993. Notes on formalizing context. In Proceedings of the 13 h International Joint Conference on Artificial Intelligence, Chambéry, France.
- McCarthy, J., S. Buvač. 1997. Formalizing context (expanded notes). In *Computing natural language*, eds. A. Aliseda, R. van Glabbeek, and D. Westerståhl. Stanford, CA: Stanford University. <http://www-formal.stanford.edu>
- Menzel, C. 1999. The objective conception of context and its logic. *Minds and Machines* 9(1):29–56, Feb 1999.

- Mitra, P., G. Wiederhold, and M. Kersten. 2000. A graph-oriented model for articulation of ontology interdependencies. Accepted for Extending DataBase Technologies, EDBT 2000, Konstanz, Germany, March 2000. <http://www-db.stanford.edu/SKC/publications.html>
- National Center for Ontological Research (NCOR). <http://ncor.buffalo.edu/>
- Neches, R., R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W.R. Swartout. 1991. Enabling technology for knowledge sharing. *AI Magazine* 12(3), Fall 1991. <http://www.isi.edu/isd/KRSharing/vision/AIMag.html>
- Noy, N.F., and M.A. Musen. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. 17th National Conference on Artificial Intelligence (AAAI-2000), Austin, TX. Technical Report SMI-2000-0831, Stanford Medical Informatics, Stanford University. http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-2000-0831.html
- Open Biomedical Ontologies (OBO). Foundry. <http://obofoundry.org>
- Object-Centered High-level Reference Ontology (OCHRE). <http://www.loa-cnr.it/DOLCE.html>
- Orbst, L. 2006. What is an ontology? A briefing on the range of semantic models. *ontolog forum*, 12 and 19 Jan 2006. http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall_2006_01_12
- Orbst, L., T. Hughes, and S. Ray. 2006. Prospects and possibilities for ontology evaluation: The view from NCOR. Workshop on Evaluation of Ontologies for the Web (EON2006), Edinburgh, UK, 22 May 2006.
- Orbst, L., and I. Mani, eds. 2000. Proceedings of the Workshop on Semantic Approximation, Granularity, and Vagueness, A Workshop of the Seventh International Conference on Principles of Knowledge Representation and Reasoning KR'2000, Breckenridge, CO, 11 Apr 2000.
- Orbst, L. and D. Nichols. 2005. Context and ontologies: Contextual indexing of ontological expressions. AAAI 2005 Workshop on Context and Ontologies, Poster, AAAI 2005, 9–13 July, Pittsburgh, PA. http://www.mitre.org/work/tech_papers/tech_papers_05/05_0903/index.html
- Orbst, L., G. Whittaker, A. Meng. 1999a. Semantic context for interoperable distributed object systems. Poster, Modeling and Using Context: Second International and Interdisciplinary Conference (Context'99), Trento, Italy, Sep 1999.
- Orbst, L., G. Whittaker, A. Meng. 1999b. Semantic context for object exchange. Workshop on Reasoning in Context for AI Applications, Patrick Brézillon, Roy Turner, Jean-Charles Pomerol, Elise Turner, co-chairs. AAAI-99, Orlando, FL, July, 1999. Technical Report WS-99-14. Menlo Park, CA: AAAI Press.
- Ontolog Forum. <http://ontolog.cim3.net/cgi-bin/wiki.pl?>
- Ontology in Information Systems (FOIS-2001), eds. C. Welty, and B. Smith, Ogunquit, Maine, 17–19 Oct 2001.
- Ontology Summit. 2007. Ontology, taxonomy, folksonomy: Understanding the distinctions. <http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2007>
- Petryzcki, L. 1955. *Law and morality*. Partial trans. by H.W. Babb, with an introduction by N.S. Timasheff. Cambridge, MA: Harvard University Press.
- Pierce, B. 1991. *Basic category theory for computer scientists*. Cambridge, MA: MIT Press.
- Poli, R. Ontology: The categorial stance. See TAO, vol 1.
- Poli, R. 1998. Levels. *Axiomathes* 9(1–2):197–211.
- Poli, R. 2001a. The basic problem of the theory of levels of reality. *Axiomathes* 12(3–4):261–283.
- Poli, R. 2001b. Alwis. Ontology for Knowledge Engineers. PhD Thesis, Utrecht.
- Poli, R. 2002. Ontological methodology. *International Journal of Human-Computer Studies* 56:639–664.
- Poli, R. 2003. Descriptive, formal and formalized ontologies. In *Husserl's logical investigations reconsidered*, eds. D. Fisette, 193–210. Dordrecht: Kluwer.
- Poli, R. 2007. Three obstructions: forms of causation, chronotopoids, and levels of reality. *Axiomathes* 16:1–18.
- Ranganathan, S.R. 1962. *Elements of library classification*, 3rd edn. New York, NY: Asia Publishing House.

- Restall, G. 2000. *An introduction to substructural logics*. New York, NY and London: Routledge.
- Rogers, J.E., and A.L. Rector. 2000. GALEN's model of parts and wholes: Experience and comparisons. Annual Fall Symposium of American Medical Informatics Association, Los Angeles, CA, 714–718. Philadelphia, PA: Hanley and Belfus Inc.
- Semy, S., M. Pulvermacher, and L. Obrst. 2005. Toward the use of an upper ontology for U.S. Government and U.S. Military Domains: An evaluation. MITRE Technical Report, MTR 04B0000063, Nov 2005. http://www.mitre.org/work/tech_papers/tech_papers_05/04_1175/index.html
- Sider, T. 2001. *Four-dimensionalism. An ontology of persistence and Time*. Oxford: Clarendon Press.
- Simons, P. 1987. *Parts: A study in ontology*. Oxford: Clarendon Press.
- Smith, B. 2001a. Fiat objects. *Topoi* 20(2):131–148.
- Smith, K., L. Obrst. 1999. Unpacking the semantics of source and usage to perform semantic reconciliation in large-scale information systems. SIGMOD special issue on Semantic Interoperability, eds. A. Sheth, and A. Ouksel, SIGMOD, Mar 1999.
- Sowa, J. 2000. *Knowledge representation: Logical, philosophical, and computational foundations*. Pacific Grove, CA: Brooks/Cole Thomson Learning.
- Suggested Upper Merged Ontology (SUMO). <http://www.ontologyportal.org/>
- Upper Cyc. <http://www.cyc.com/cycdoc/vocab/vocab-toc.html>
- Uschold, M., M. Gruninger. 1996. Ontologies: Principles, methods, and applications. *The Knowledge Engineering Review* 11(2):93–136.
- Varzi, A. 2000. Vagueness, logic, and ontology, to appear in *The Dialogue*. http://www.columbia.edu/~av72/papers/Dialogue_2000.pdf
- Varzi, A.C. 1998. Basic problems of mereotopology. In *Formal ontology in information systems*, eds. N. Guarino, 29–38. Amsterdam: IOS Press.
- Varzi, A., and F. Pianesi. 1996a. Events, topology, and temporal relations. *The Monist* 78(1): 89–116.
- Varzi, A., and F. Pianesi. 1996b. Refining temporal reference in event structures. *Notre Dame Journal of Formal Logic* 37(1):71–83.
- Wiederhold, G. 1994. An algebra for ontology composition. Proceedings of 1994 Monterey Workshop on Formal Methods, Sept 1994, U.S. Naval Postgraduate School, Monterey, CA, 56–61. <http://www-db.stanford.edu/pub/gio/paperlist.html>
- Williamson, T. 1998. *Vagueness*. London and New York, NY: Routledge.

Chapter 2

Ontological Architectures

Leo Obrst

2.1 Introduction

We distinguish between *ontological architecture* and *ontology architecture*, though they are closely related. *Ontology architecture* is emerging as a distinct discipline in ontology engineering – as an ontology development and deployment structure and methodology (Fernandéz et al., 1997). It necessarily also includes aspects of what is sometimes termed *ontology lifecycle management* (Andersen et al., 2006). In fact, ontology architecture can be considered to encompass ontology lifecycle management because the former lays out a general framework for the development, deployment, and maintenance of ontologies (which is the focus of lifecycle management), but also includes the interaction of applications and services that use ontologies, and an ontology tool and service infrastructure to support these. *Ontological architecture* is the architecture that is used to structure the ontologies that are employed by ontology architecture. As such, it addresses the levels of ontologies required (foundational, upper, middle, utility, reference, domain, and sub-domain ontologies), and mathematical, logical, and engineering constructs used to modularize ontologies in a large *ontological space*. This chapter focuses on *ontological architecture*, but it must be understood to underpin *ontology architecture* if only to ground/situate and enable the latter. Both kinds of architecture are relevant to ontology engineering, but we cannot address ontology architecture here until the very last section, when we look ahead. Instead, we focus on ontological architecture, which as it turns out, is a large enough topic.

The chapter is structured as follows. In Section 2.2, we distinguish *ontological architecture* from *ontology architecture*, provide some understanding of their respective rationales, how ontologies are distinct from but impinge on elements of epistemology, the formal semantics of language, and conceptual models. We depict the *ontology spectrum* (Obrst, 2002–2003), which constitutes a range of semantic models of increasing expressiveness, and define these. The more expressive

L. Obrst (✉)
The MITRE Corporation, McLean, VA, USA
e-mail: lobrst@mitre.org

models enable more complex applications. We also show how one aspect of ontological architecture, the expressiveness of the knowledge/ontology representation language, and ontology application are related. In Section 2.3, ontological architecture is described, by detailing the use of upper, middle, and domain ontologies to address semantic interoperability. We extend this with a discussion of additional structure that has been proposed, and some foundational ontological distinctions. Section 2.4 is the core of the chapter. It discusses some ways of structuring the *ontological space*, which really is itself embedded in a *logical space*, and necessarily must also address *meta-ontological architectural* issues. Notions of ontological modularity are examined, including that of formalized contexts such as *microtheories*, which originated from the Cyc effort (Blair et al., 1992), the approach called *the lattice of theories*, most recently characterized by John Sowa (2005), additional approaches based on logical ways of characterizing mathematical *little theories* (Farmer et al., 1992) which yet must interoperate, recent research in ontology modularity, and Robert Kent's (2004, 2006) meta-ontology called *the Information Flow Framework*, based on Barwise and Seligman's (1997) *Information Flow Theory*, itself an application of Category Theory (Mac Lane, 1971; Bar and Wells, 1999), and similar work at the meta-ontological level. Finally, in Section 2.5, we conclude with a vision of the future for both ontological and ontology architecture.

Ontological architecture spans many topics. We can only briefly sketch its components in this chapter.

2.2 Ontological and Ontology Architecture: Overview

Ontology architecture addresses content (how better ontologies are developed), the apparatus needed to develop, deploy, and maintain ontologies (which tools, requirements, methodologies, lifecycle support, policy, and governance are required), and ontology application interaction (how data is stored, accessed, and linked to ontology instances and facts; which services do ontology require for and provide to applications). We do not address this ontology architecture per se in this chapter, since our interests are more fundamental. Instead, we focus on *ontological architecture*, the foundational architecture which must underpin subsequent ontology application notions we characterize as ontology architecture. This section provides an overview of what ontological architecture addresses.

2.2.1 Truth and Belief: Ontology, Epistemology, Contextual Semantics, Language, and Applications

Ontology is many things to many people, as the other chapters of these volumes demonstrate, and so no time is spent here defining ontology. This chapter focuses on architecture. One issue, however, needs to be raised: what ontology does not address

particularly must still be addressed by ontology architecture. Ontology is not epistemology, nor is it the semantics of natural language. But aspects of these must be addressed by an account of ontology architecture. Why *epistemology*? Because, though ontology is about the real entities, relations, and properties of the world, epistemology is about the perceived and belief-attributed entities, relations, and properties of the world, empirical evidence gleaned that will be described or characterized by ontology. Why *natural language semantics*? Because, though ontology is about the real entities, relations, and properties of the world, natural language semantics is about the rendition in language of interpretations about the entities, relations, and properties of the world, and includes notions of sense and reference. In ontology architecture, epistemology is employed in the use and qualification of data and as stored in databases or tagged or indexed in documents. In ontology architecture, natural language semantics is employed in the analysis of natural language descriptions used to ascertain and represent the real world entities of ontology, the naming conventions used and the access to the interpretations about the real world that the ontology represents. One natural language processing technology in particular, information extraction, crucially depends on natural language semantics -- information extraction addressing the identification of entities, relations, and events in unstructured text, and the tagging or extraction of these to form instances of ontology concepts.

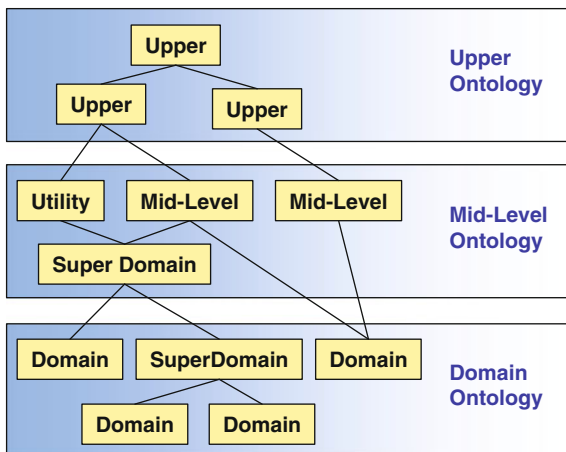
2.2.2 The Big Picture

Figure 2.1¹ is a graphical rendition of ontological architecture and its components. These components will be described in more detail in Section 2.3.

The important point about this diagram is the layers: the upper, mid-level, and domain (or lower) ontology layers. Sometimes the upper and mid-level ontologies are called foundational ontologies, and there can be multiple such in each layer. We eschew the notion of a monolithic ontology, at least for engineering purposes, and instead view foundational ontologies similar to domain ontologies: as coherent and consistent theories, hence our use of the terminology introduced in the next section, i.e., ontologies as *logical theories*. In our view, upper ontologies are most abstract and make assertions about constructs such as identity criteria, parts/wholes, substance and constitution, space and time (and endurants and perdurants), necessary properties, dynamic properties, attributes spaces, etc., that apply to all lower levels; hence, they span all mid-level and domain ontologies. Upper ontologies themselves may consist of levels, as the extended discussions on *levels of reality* make clear (Poli, 2003; Poli, 2010, this volume; Poli and Obrst, 2010, this volume). Mid-level ontologies are less abstract and make assertions that span multiple domain ontologies. The characterization of these are less clear, as is the demarcation point between upper and mid-level. Some examples of constructs potentially in

¹See Semy et al. (2004, p. 8), also Chapter 1 by Poli and Obrst, this volume.

Fig. 2.1 Ontology architecture: the big picture



a mid-level ontology are *humanOrganization* and *industrialProcess*. These are not necessarily represented in an upper ontology, but may be; probably, however, they are in a mid-level ontology (a biomedical mid-level ontology may not have these as concepts; however, a manufacturing mid-level ontology would have them).

2.2.3 The Ontology Spectrum

Ontology architecture, just like ontology, is a notion that must be learned and incorporated gradually over time by an enterprise or community. It's possible, though rare, that an enterprise or community is sufficiently knowledgeable about ontology and its use in semantically informing applications; it's equally rare that the organization is aware of the costs, hazards, cost-effective benefits, and preferred methods of using ontology. To assist organizations (enterprises, communities) in determining the range of semantic models that constitute stages of embracing semantic technologies of which the highest model is ontologies, we have created the Ontology Spectrum, depicted in Fig. 2.2 (Obrst, 2002; Obrst, 2003; Daconta, Obrst, Smith, 2003). The Ontology Spectrum is both a description of the range of semantic models, as they increase in expressivity and complexity of structure, and an indication of the migration path that organizations can take as the expressiveness of their semantic models needs to increase in order to accommodate their richer problems and hence, solutions. The lower end of the Ontology Spectrum is in fact not about ontologies at all.

What is colloquially, though incorrectly, known as an *ontology* can range from the simple notion of a *Taxonomy* (terms² or concepts with minimal hierarchic or

²We differentiate and define *term* and *concept* in Table 2.1, below.

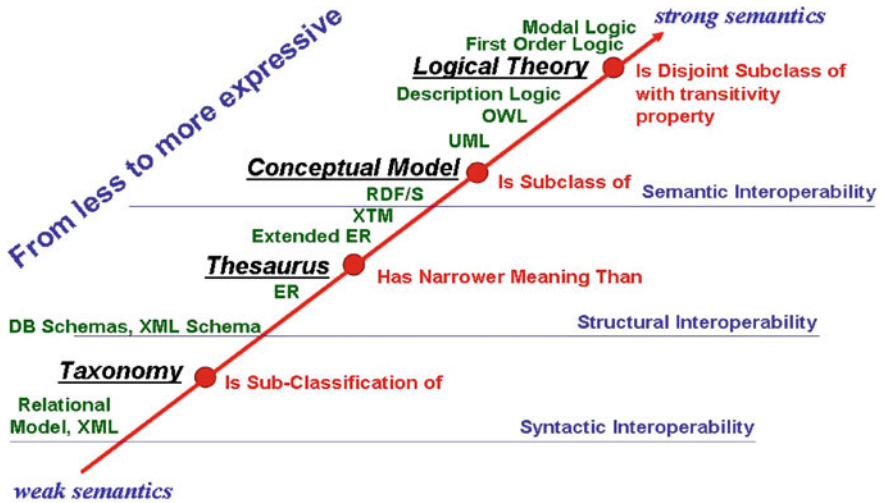


Fig. 2.2 The ontology spectrum

parent/child structure), to a *Thesaurus* (terms, synonyms, broader than/narrower than term taxonomies, association relation), to a *Conceptual Model* (concepts structured in a subclass hierarchy, generalized relations, properties, attributes, instances), to a *Logical Theory* (elements of a Conceptual Model focusing however on real world semantics and extended with axioms and rules, also represented in a logical KR language enabling machine semantic interpretation). Terms and concepts are differentiated in Table 2.1. We also differentiate between weak and strong taxonomies: The *subclassification of relation* characterizes *weak taxonomies*. *Strong taxonomies* are characterized by either the *subclass of* relation for concepts (which typically can be considered universal categories for referents³) or the *narrower*

³There is, however, a vast literature on the notion of *concept* in philosophy, cognitive science/psychology, and linguistics. Often in cognitive science/psychology, a concept is considered to be mental particular, having a structured mental representation of a certain type (Margolis and Laurence, 1999b, p. 5–6). However, in philosophy, a concept is considered an abstract entity, signifying a general characterizing idea or universal which acts as a category for *instances* (*individuals* in logic, *particulars* in metaphysics and philosophical ontology) (Smith, 2004). Even in the philosophical literature, the notion of concept will vary according to philosophical stance, i.e., according to whether the adherent to the particular notion is an idealist, nominalist, conceptualist, or realist, or some combination or refraction of those (Poli, 2010). For example, some will consider a concept to be simply a placeholder for a real world entity, either a universal or a particular; example: *Joe Montana* (a former USA football quarterback) or *Winston Churchill* (a former UK prime minister) as concepts. That is, the mental placeholder or idea can be about anything. This notion of concept is a surrogate for anything that a philosophical or many linguistic theories may opine. Often, therefore (and this is our view here), concepts are best understood as *conceptions*, a term which has perhaps less technical baggage, insofar as *conception* emphasizes that we are talking about a mental representation which may or may not be reified as a *concept*, perhaps a stronger notion. But

Table 2.1 Term vs. concept

Terms (terminology): Natural language words or phrases that act as indices to the underlying meaning, i.e., the concept (or composition of concepts). The term is syntax (e.g., a string) that stands in for or is used to indicate the semantics (meaning).

Concept (a universal category for referents): A unit of semantics (meaning), the node (entity) or link (relation) in the mental or knowledge representation model. In an ontology, a concept is the primary knowledge construct, typically a class, relation, property, or attribute, generally associated with or characterized by logical rules. In an ontology, these classes, relations, properties are called concepts because it is intended that they correspond to the mental concepts that human beings have when they understand a particular body of knowledge (subject matter area or domain) but at the philosophical *universal* level, i.e., as *kinds* of entities. In general, a concept can be considered a placeholder for a category (way of characterizing) of specific real world referents (synonymously: specific entities, instances, individuals, or particulars), and thus ontology as an engineering product is about representing the semantics of the real world in a model that is usable and interpretable by machine.

than relation (thesauri) for terms. Only the subclass/narrower than relation is a *generalization-specialization* relation (subsumption).⁴

A Conceptual Model can be considered a *weak ontology*; a Logical Theory (Fig. 2.3) can be considered a *strong ontology*. The innermost circle is the set of axioms. The middle circle is the set of theorems. The outermost circle is the ever expanding theory, an ontology as logical theory about reality which grows over time, as new axioms are entered and new theorems deduced. An ontology as a logical theory is thus: (1) a set of (non-logical) *axioms*, i.e., the classes, properties, subclass and subproperty assertions, the relations, attributes, and constraints on these; (2) the potentially expanding set of *theorems*, which can be proven true by some valid justification mechanism such as that which a typical formal logic provides, i.e., a set of equivalences or valid reasoning patterns known as *inference rules*, e.g., Modus Ponens; (3) *interpretations*, which are not depicted in the figure, are the mappings between a given theory and the set of models (in the sense of *model-theory* (Makowsky, 1992; Van Leeuwen, 1994; Hodges, 1997), which are supposed to be what the syntactic expressions of the theory *mean*. The whole, a logical theory, constitutes the specific, growing ontology.

The primary distinction here between a weak and a strong ontology is that a weak ontology is expressed in a knowledge representation language which is not

for purposes of simplicity, we use the term *concept* in this chapter to mean roughly *an abstract entity signifying a general characterizing idea or universal which acts as a category for instances*.

⁴The *subsumption* relation is typically defined to be the *subset* relation, i.e., intuitively a class is similar to a set, and the instances of that class are similar to elements of the set. A more general class (set), therefore, like *mammal* will contain a subclass (subset) of *primate*, among whose instances (elements) will be specific humans like Ralph Waldo Emerson. *Concept subsumption* as an ontology reasoning problem means that “given an ontology *O* and two classes *A*, *B*, verify whether the interpretation of *A* is a subset of the interpretation of *B* in every model of *O*” (OWL 1.1. <http://www.w3.org/Submission/owl111-tractable/>).

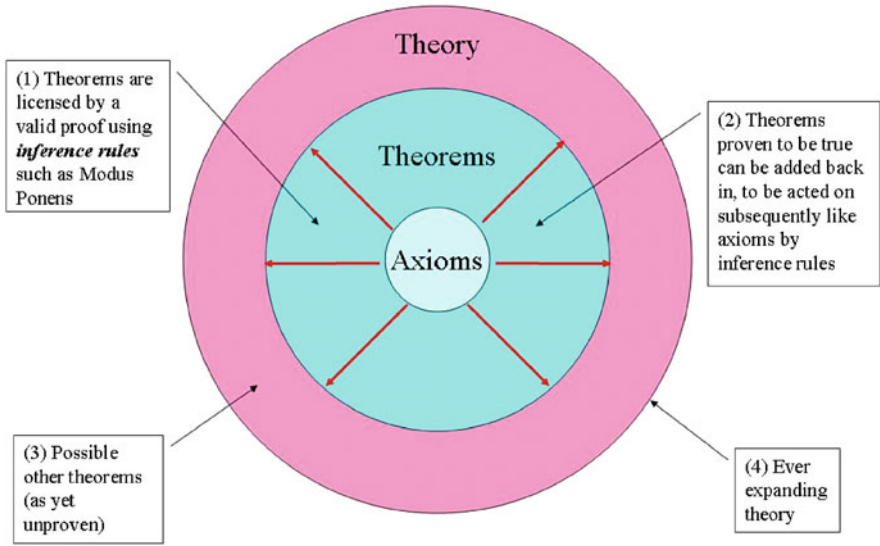


Fig. 2.3 Ontology as logical theory

based on a formal logic. Why is this important? It means that a machine can only read and process a weak ontology (e.g., currently models in ER or UML). It cannot semantically interpret the ontology, i.e., ingest the ontology and perform automated reasoning on it (reasoning which is similar to that which a human would make). So a weak ontology is not semantically interpretable by machine; a strong ontology is.

So what is usually, colloquially considered by the larger community to be an ontology needs clarification: all of these models should instead be considered *semantic models*. An ontology is restricted to the upper half of the Ontology Spectrum The Ontology Spectrum therefore displays the range of models in terms of expressivity or richness of the semantics that the model can represent, from “weak” or less expressive semantics at the lower left (value set, for example), to “strong” or more expressive semantics at the upper right. The vertical lines, labeled by *syntactic interoperability*, *structural interoperability*, and *semantic interoperability*, indicate roughly the expressiveness of the model require to respectively address those levels of interoperability.⁵ *Syntactic interoperability* is defined as enabling the interchange of information based on a common syntax for at least that interchange. *Structural interoperability* is defined as a providing a common structure (a higher-order syntax) to enable the interchange of information. For example, multiple documents

⁵There are both lower levels of interoperability and higher levels. Lower levels include logical and physical accessibility and connectivity interoperability, e.g., having two information sources on the communication network, with network addresses known by those who might wish to access those sources. A higher level might be *pragmatic interoperability* (intending a formal pragmatics account), which factors in the intent of the represented semantics.

may be syntactically represented in XML, but need to be validated against distinct structural XML schemas or Document Type Definitions (DTD), which can be viewed as grammar rules that organize components of the syntax in specific ways. *Semantic Interoperability* is defined as providing a common semantics to enable the interchange of information, i.e., the semantics of the structural layer: what those structural components mean.

As depicted in the Ontology Spectrum, XML is sufficient for syntactic interoperability, XML Schema enables structural interoperability, but a minimum of RDF is necessary for semantic interoperability.

Figure 2.4 maps those semantic models against the increasingly more complex applications that are able to be addressed by using those models.

In the above diagram, *term* (terminology) and *concept* (real world referent) are defined as previously in Table 2.1.

As the expressiveness of the semantic model increases, so does the possibility of solving more complex problems. At the Taxonomy level, an application can provide only simple categorization, indexing, search, navigation: for example, indexing your documents into loose topic buckets with some hierarchic organization. Using thesauri can enable a search application to increase recall by, for example, using synonyms and substituting these into an expanded query string. For applications that require more precision, i.e., where approximate or loose characterizations of the semantics simply will not accomplish what is needed, more expressive models such as Conceptual Models and Logical Theories, i.e., ontologies, are required.

Recall is a measure of how well an information search and retrieval system finds ALL relevant documents on a searched for topic, even to the extent that it includes some irrelevant documents. *Precision* is a measure of how well such a system finds

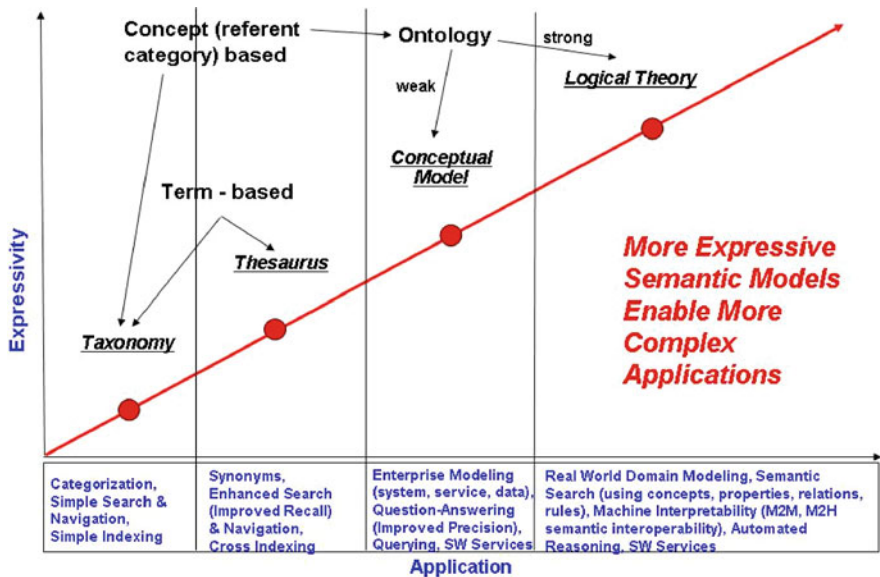


Fig. 2.4 More expressive semantic models enable more complex applications

Table 2.2 Recall vs. precision⁶

Recall: The percentage of relevant documents retrieved:
$\frac{\text{Number of relevant docs retrieved}}{\text{Number of relevant docs}}$
Precision: The percentage of retrieved documents judged relevant:
$\frac{\text{Number of relevant docs retrieved}}{\text{Number of docs retrieved}}$

ONLY relevant documents on a searched for topics, even to the extent that it skips irrelevant documents. Table 2.1 displays the usual definitions of recall and precision. In most cases, recall and precision are inversely proportional to one another, with high recall leading to low precision, and high precision meaning the recall is low (Buckland and Gey, 1994; Table 2.2).

2.2.4 The Ontology Maturity Model

Building on the notions of the Ontology Spectrum, we describe one possible view of how an enterprise may migrate from less expressive semantic models to more expressive models, i.e., to real ontologies, based on both the common understanding of the enterprise and its requirements for more complex applications. Figure 2.5 displays an overall Ontology Maturity (or Capability) Model, simplified here, that shows the significant gradations toward greater maturity an organization may take in its evolution toward more completely realizing the goal of an ontology-driven enterprise.

This figure, which is patterned after the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) that was intended to describe and gauge an organization's software process maturity (Royce, 2002), we attempt to develop a scale of maturity in an organization's migration towards increasingly more robust approaches to the use of ontologies for information technology needs.

Our analysis is that initially an organization thinks primarily of local semantics, i.e., attempts to characterize their information technology needs based on (currently mainstream) syntactic and structural methods, with only implicit semantics: a nodding of the head to signify agreement with the semantics as uttered in speech, or an agreement on a data dictionary of English or other natural language definitions, which ostensibly humans can read and indirectly nod their heads over. However, as an organization evolves, it begins to understand that it is actually composed of many communities and sub-organizations, each of which has its own local semantics but in addition a common enterprise-wide semantics, in fact a common semantics based

⁶“Recall is like throwing a big fishing net into the pond. You may be sure to get all the trout, but you've probably also pulled up a lot of grouper, bass, and salmon, too. Precision is like going spear fishing. You'll be pretty sure to ONLY get trout, but you'll no doubt miss a lot of them, too.” – Jim Robertson, <http://www-ec.njit.edu/~robertso/infosci/recall-precision.html>

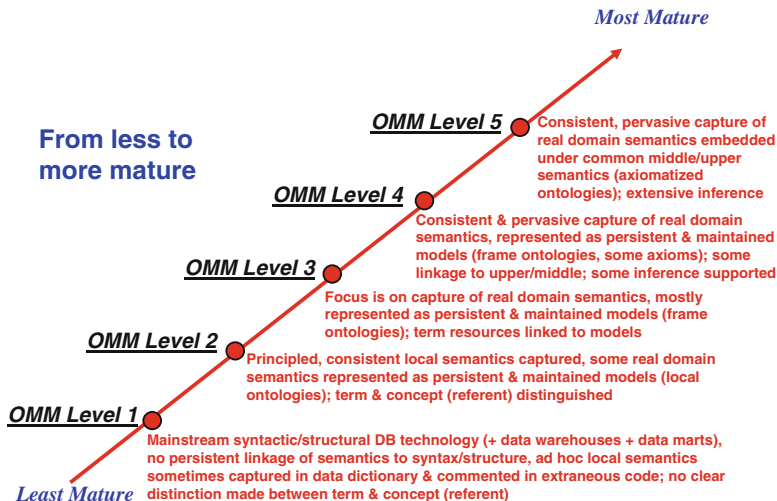


Fig. 2.5 Ontology maturity model (OMM)

on real world referents that all communities and sub-organizations in the enterprise share. Most commonly, as a semantically aware enterprise matures, it eventually distinguishes between terms (ways of referring) and concepts/referents (referents referred to by potentially many different terms). Hence, the semantic models the maturing enterprise embraces evolves from term-based models (weak taxonomies and thesauri) to concept/referent-based models (weak and strong ontologies).

In addition, as the maturing enterprise begins to understand that terminologies are not as necessary as the underlying meanings (concepts) of those terminologies that get modeled as a machine usable or interpretable engineering semantic model (ontology), the enterprise tries to fit together the local semantic models it currently has (local database schemas or even local community ontologies). Because it is soon recognized that there is great and incommensurable, though seemingly duplicative, meaning among the diverse ontologies (conceptual stovepipes), the enterprise attempts to reconcile the semantics. It does so initially by trying to construct semantic mappings between the two ontologies, and then when the problem repeats itself with every additional ontology which needs to be incorporated (mapped to), the enterprise begins to understand that the emerging mapping ontology is actually an integrative ontology that must be as expressive as the most expressive of the ontologies needing to be integrated.

2.3 Ontological Architecture: Upper, Mid-level, Domain Ontologies

In this section we discuss the fundamentals of ontological architecture. As depicted in Fig. 2.1, an ontological architecture encompasses primarily three layers: upper ontologies, mid-level ontologies, and domain ontologies, with the first two also

sometimes called *foundational ontologies*. This section focuses on these. However, in Section 2.4, we will generalize the architecture to include a meta-level.

2.3.1 What Is an Upper Ontology?

Ontologies may exist at many levels of abstraction. We group ontologies into three broad categories of upper, mid-level and domain ontologies. In this section we define what we mean by an upper ontology and characterize the differences between these three levels. Figure 2.6 is a graphical depiction of these notional levels along with some sample concepts that may be found at each level.

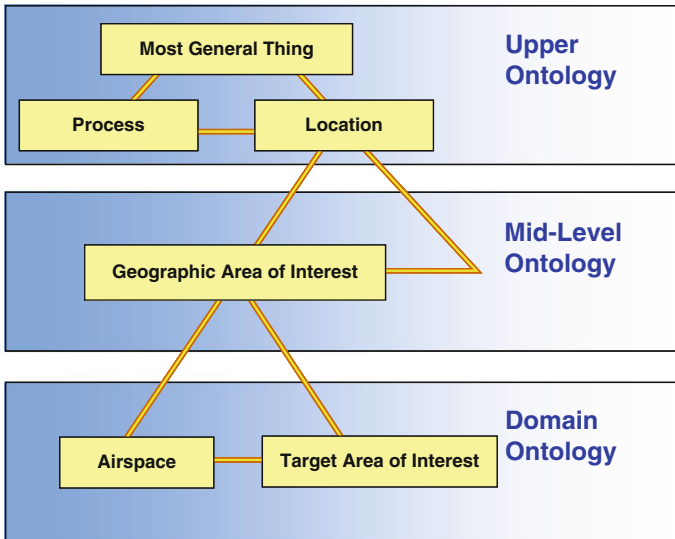


Fig. 2.6 Ontology categories

2.3.1.1 Upper Ontology Definition

An upper ontology, as defined by Phylita (2002), is a high-level, domain-independent ontology, providing a framework by which disparate systems may utilize a common knowledge base and from which more domain-specific ontologies may be derived. The concepts expressed in such an ontology are intended to be basic and universal concepts to ensure generality and expressivity for a wide area of domains. An upper ontology is often characterized as representing common sense concepts, i.e. those that are basic for human understanding of the world (Kiryakov et al., 2001). Thus, an upper ontology is limited to concepts that are meta, generic, abstract and philosophical.⁷ Standard upper ontologies are also sometimes referred to as foundational ontologies⁸ or universal ontologies (Colomb, 2002).

⁷Standard Upper Ontology (SUO) Working Group Website, <http://suo.ieee.org/>.

⁸OpenCyc Website, <http://www.opencyc.org/>.

2.3.1.2 Upper Ontology vs. Mid-Level Ontology

A mid-level ontology serves as a bridge between abstract concepts defined in the upper ontology and low-level domain specific concepts specified in a domain ontology. While ontologies may be mapped to one another at any level, the mid-level and upper ontologies are intended to provide a mechanism to make this mapping of concepts across domains easier. Mid-level ontologies may provide more concrete representations of abstract concepts found in the upper ontology. This ontology category also encompasses the set of ontologies that represent commonly used concepts, such as Time and Location. These commonly used ontologies are sometimes referred to as utility ontologies.

2.3.1.3 Upper Ontology vs. Domain Ontology

A domain ontology specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective. While the same concept may exist in multiple domains, the representations may widely vary due to the differing domain contexts and assumptions. Domain ontologies may be composed by importing mid-level ontologies. They may also extend concepts defined in mid-level or upper ontologies. Reusing well established ontologies in the development of a domain ontology allows one to take advantage of the semantic richness of the relevant concepts and logic already built into the reused ontology. The intended use of upper ontologies is for key concepts expressed in a domain ontology to be derived from, or mapped to, concepts in an upper-level ontology. Mid-level ontologies may be used in the mapping as well. In this way ontologies may provide a web of meaning with semantic decomposition of concepts. Using common mid-level and upper ontologies is intended to ease the process of integrating or mapping domain ontologies.

2.3.2 Why Do We Care About Upper Ontology?

2.3.2.1 How Upper Ontologies May Help

Today's World Wide Web (WWW) is geared toward presenting information to humans. The Semantic Web is an evolution of the WWW that is intended to capture the meaning of data (i.e., data semantics) precisely enough that a software application can interpret them. A key element of the Semantic Web is the use of ontologies to define concepts and their relationships. With ontologies supplying the context of data, information retrieval and search engines can exploit this contextual information to perform semantic searches based on the meaning of the concept, rather than syntactic searches of a given text string. In this way, one could discriminate between horses and cars which both have the same label of "mustang." Rich semantics captured in ontologies also provide the ability to combine simple facts together to infer new facts, and to deduce new generic knowledge in the form of

proven theorems that is only implicit in the ontologies. With data and applications mapped to ontologies, inference engines could be used to improve the discovery and understanding of data as well as the discovery and composition of applications like Web services. Furthermore, ontologies may be used to represent the semantics of applications and services directly, much as UML object and conceptual models do today for specific systems and enterprises, though these do so incompletely, inconsistently, and unsoundly, without explicit use by the applications of these models at either system-generation time or run-time. Upper ontologies are intended to define foundational concepts used in both mid-level and domain ontologies. In theory, the mapping between domain ontologies becomes easier if the ontologies to be mapped are derived from a standard upper ontology.

Two approaches exist for the use of upper ontologies: top-down and bottom-up. In a top-down approach one uses the upper ontology as the foundation for deriving concepts in the domain ontology. In this way, the domain ontology designer takes advantage of the knowledge and experience already built into the upper ontology. Furthermore, use of the upper ontology provides a theoretical framework on which to build. In a bottom-up approach, the ontology designer maps a new or existing domain ontology to the upper ontology. This approach also capitalizes on the knowledge built into the upper ontology but one would expect the mapping to be more challenging, as inconsistencies may exist between the domain and upper ontology. Some upper ontologies utilize a combination of these two approaches.

2.3.2.2 A Software Engineer Analogy

Let's use a software engineering analogy to describe the value of using standard upper and mid-level ontologies. Mid-level ontologies can be seen as analogous to software libraries. Early high level programming languages evolved to contain software libraries of commonly used functions. High quality software libraries allowed programmers to reuse the knowledge and experience built into the software library and freed them to concentrate on domain specific issues. As software libraries evolved, programming tasks became easier. Programmers do not need to understand the detailed implementation of libraries in order to use them. Similarly, mid-level ontologies can evolve to act as ontological utilities. With the existence of such ontologies, ontology designers can compose their domain ontologies using these utility ontologies and inherit the concepts and inferencing capabilities provided by them. Just as software libraries make programming tasks easier, so too would the availability of high quality, commonly used utility ontologies make ontology development easier. Further, concepts in the utility ontology could be mapped to concepts in an upper ontology without the need for users of the utility ontology to be aware of these mappings.

Because it is early in the Semantic Web evolution (OWL became a World Wide Web Consortium [W3C] recommendation in Feb'04), few utility ontologies exist. However, they are emerging, as evidenced by the DARPA funded effort to create a standard time ontology, now a W3C public working draft (Hobbs and Pan, 2006).

2.3.3 What Foundational Ontologies Provide: Ontological Choices

We cannot evaluate foundational ontologies here (but see Section 2.3.4). However, we can provide some rationale for why foundational ontologies are useful in an overall ontological architecture and what kinds of constructs one might desire for a foundational ontology. We call these *ontological choices* (though Partridge (2002) calls them *meta-ontological choices*).

What are the ontological choices that a given foundational ontology provides? These ontological choices will entail *ontological commitments*, which means that there is downward impact on mid-level and domain ontologies on the decisions one makes at the upper or foundational levels. The WonderWeb Ontology Library Final Report (Masolo et al., 2003), for example, describes a number of such ontological choices: descriptive vs. revisionary, multiplicative vs. reductionist, universals vs. particulars vs. sets, endurants vs. perdurants, and more. Other choices include 3-dimensional (3D) vs. 4-dimensional (4D) (Hamlyn, 1984; Loux, 2002), distinct notions of “part” and “whole”, different notions about what constitutes a property (and attribute), how change should be represented, distinctions about granularity, vagueness, etc.⁹ Many of these choices are intricately linked, so, for example, discussions on endurants and perdurants invoke 3D and 4D views, and crucially elucidate the notion of persistence through time and change. In addition, multiplicative ontologies, because they tolerate a greater range of modeling complexity (model whatever is called for by reality), generally enable multiple objects with different identity criteria to co-occur/co-locate in the same spacetime (Masolo et al., 2003). In the following, we discuss some of these choices.

2.3.3.1 Descriptive vs. Revisionary

Descriptive and *revisionary* ontologies (Strawson, 1959) are based on ontological stances or attitudes towards the effort of modeling ontologies, i.e., how one conceptualizes the world and what an ontological engineering product is or should be. A *descriptive* ontology tries to capture the more commonsensical and social notions based on natural language usage and human cognition, emphasizing the agent who conceives and deemphasizing scientific and philosophical considerations. A *revisionary* (sometimes called *prescriptive*) ontology, on the other hand, does emphasize (or even, strictly adheres to) the scientific and philosophical perspectives, choosing to base its constructs and modeling decisions on scientific theories and a philosophical stance that tries to capture the world as it really is (it *prescribes* the world), and not necessarily as a given historical agent conceives it to be. A revisionary ontology therefore says that its modeling constructs are about real things in the world as it is.

⁹Another choice we will not investigate here is that between *presentism* and *eternalism* (Partridge, 2002). Presentism argues that time is real; eternalism that time is not real, that entities change but their properties do not change over time. Presentism typically goes with endurantism; eternalism goes with perdurantism.

In practical terms, all of the constructs in a *revisionary* ontology will be space-time objects, i.e., necessarily having temporal properties; in a *descriptive* ontology, that will not be the case. In the latter, *entities* (sometimes called *endurants*, but perhaps better called *continuants*) such as “hammer” and “tank” that have only incidental temporal properties and *events* (processes, actions, activities, etc., sometimes called *perdurants*, but perhaps better called *occurrents*) such as “attacking” and “cashing a check” that have explicit temporal properties, are modeled with or without those temporal properties, respectively. Often in natural language there are two correlated forms/usages that express the distinction: the nominal and the verbal. A nominal (noun) “attack” is expressed as in “The attack on the enemy began at 600 hours.” A verbal (verb) “attacked” is expressed as in “We attacked the enemy at 600 hours.”

2.3.3.2 Multiplicative vs. Reductionist

A multiplicative upper ontology is expressively profligate in that concepts can include anything that reality seems to require, and so any distinction that seems useful to make can be made in the ontology. Contrarily, a reductionist ontology reduces the number of concepts to the fewest primitives sufficient to derive the rest of complex reality.

In the WonderWeb Foundational Library (Masolo et al., 2003), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) and the Basic Formal Ontology (BFO) are multiplicative and descriptive, whereas the Object-Centered High-Level Reference Ontology (OCHRE) is reductionist and revisionist. The Suggested Upper Merged Ontology (SUMO)¹⁰ (Niles and Pease, 2001b) could be said to be both multiplicative in that it aims to cover at a general level any concept that reality requires, and reductionist in that it attempts to be minimal rather than profligate.

We note that many of these dichotomous ontology choices (*descriptive* vs. *revisionary*, *multiplicative* vs. *reductionist*, etc.) really have behind them a set of assumptions about how to view the world (e.g., strict realism with no notion of a different possibility) and what an engineering model of the world or parts of the world can achieve. Therefore, many of the ontology choices will tend to co-occur: e.g., *revisionist* and *reductionist* will generally go together.

2.3.3.3 Universals, Particulars, Sets, Possible Worlds

The distinction between *universals* (forms, ideas) and *particulars* (individuals) brings up a range of philosophical argument that we cannot address here. For our purposes, universals (whether based on realism, conceptualism, or nominalism) are general entities. Universals are often characterized as natural classes that abstract or generalize over similar particular things. Person, Location, Process, etc., are

¹⁰Suggested Upper Merged Ontology (SUMO) Website. <http://www.ontologyportal.org/>.

	Level	Example Constructs	
Meta-Level to Object-Level	Knowledge Representation (KR) Language (Ontology Language) Level: Meta Level to the Ontology Concept Level	Class, Relation, Instance, Function, Attribute, Property, Constraint, Axiom, Rule	Language
	Ontology Concept/Type (OC) Level: Object Level to the KR Language Level, Meta Level to the Instance Level	Person, Location, Event, Frog, non-SaccharomycesFungusPolarize dGrowth, etc.	Ontology (General)
Meta-Level to Object-Level	Instance (OI) Level: Object Level to the Ontology Concept Level	Harry X. Landsford III, Person560234, Frog23, non-SaccharomycesFungusPolarize dGrowth822,	Knowledge Base (Particular)

Fig. 2.7 Ontology representation levels (Obrst et al., 2007)

examples of universals, and would be represented at the Ontology Concept Level in Fig. 2.7 (see next section).

If you take a *realist* stance, universals are “entities of a general kind that exist independently of us and of our ways of thinking and speaking about the world” (Hamlyn, 1984). A *conceptualist* views universals as existing in human minds and primarily functioning as concepts that generalize and classify things. *Nominalists* view universals as largely a notion of our human language, the mode of expression of our thoughts. In an extreme view of realism, Platonism, universals independently exist (it’s usually considered unproblematic that particulars exist in reality), and so in our discussion of upper ontologies here, universals would exist in a quantificational domain distinct from that of particulars. This could be the case, for example, if universals were represented at the Ontology Concept level, but the Knowledge Language level of Fig. 2.7 permits second-order quantification, i.e., quantification over concepts (properties, predicates, classes, relations, etc.), rather than just over particulars (individuals, instances) at the Ontology Instance level.

A further distinction can be made: some instances (particulars or individuals) can themselves be considered universals – at least from the perspective of ontology applications (Welty, Ferucci, 1999). Degen et al. (2001) address this issue by introducing universals of higher-order. The Semantic Web ontology language OWL in fact allows for classes as instances in the OWL-Full dialect (Smith et al., 2004).

Particulars, or individuals or instances, are specific entities and taken to be instantiations of universals. Particulars exemplify properties (which are usually understood as universals), meaning they possess specific values such as Sam Jones being the father of Bill Jones, this apple in my hand being red, and that ball being on that table at 11 am EST, on January 19, 2008, in my house in Fairfax, Virginia, USA. Particulars are represented at the Instance Level in Fig. 2.7. Instances of classes (concepts), *facts* (specific instantiated relations/properties, e.g., Sam’s fatherhoodness to Bill, my apple’s redness), and *events* (a fact that occurs at a specific time, a specific perdurant) (Pianisi and Varzi, 2000; Higginbotham et al., 2000) are typically taken to be particulars.

Sets are mathematical objects that are sometimes, but not always used to abstractly characterize the different ontological categories, i.e., the logical apparatus used to define and order the logico-mathematical notions of ontology. Model-theoretical semanticists use set theory, but formal ontologists sometimes object (see e.g., Smith, 2001), where *mereotopology* (discussed below) is argued to provide a better foundation for ontology. Nonetheless, a set does not typically constitute a separate ontological category in its own right – except insofar as it is used as a human artifact. So, for example, SUMO defines a *set* as an ontological entity in its upper ontology because it does represent an entity that it used by other components of the SUMO upper ontology and potentially other lower, domain ontologies which use SUMO and make reference to sets directly, as ontological objects. A set in the first sense, i.e., as a defining mathematical notion, would typically be expressed at the meta-level, i.e., the Language Level in Fig. 2.7, and thus is not itself an object for ontological modeling.

It is perhaps a bit confusing or disconcerting to find that the object *set* really exists at two levels, i.e., at the modeling content level (Concept Level in Fig. 2.7) and also at its meta-level (Language Level, Fig. 2.7). The confusion devolves at least partially on the distinction between *use/mention* (Tarski, 1933, 1944), i.e., natural language typically allows one to both use a word and to mention it. So in this sense, ‘set’ is both an ontological object at the Ontology-Concept modeling level, and the meta-level object at the Language level which helps to define the entire Ontology-Concept level below it.

An additional consideration – which we will not discuss in any detail here – is the notion of *possible worlds*, which is a way of formally characterizing the distinction between *descriptions* (intensions) and *individuals which possess the properties described by the descriptions* (extensions). In a sense, the Cyc context and *microtheory*-based systematic manner of segregating assertions into theories, two of which taken together and compared may contradict each other, can be considered an implementation of the notion of possible worlds. *Possible worlds semantics* is usually a notion that also involves modal logic. We consider these notions in more detail in Section 2.4, Structuring the Ontological and Meta-Ontological Space.

2.3.3.4 Endurants and Perdurants

The distinction between *endurants* and *perdurants* is sometimes conflated with two different distinctions: (1) the distinction between *3D* and *4D* ontological objects, and (2) the distinction between *continuant* and *occurrent*, respectively. However, these confluations are problematic (Hayes et al., 2002; Sider, 2004; Degen et al., 2001). According to the usual definitions (Bittner and Smith, 2003), an *endurant* is an entity which exists in full in every instant at which it exists at all; a *perdurant* “unfolds itself over time in successive temporal parts or phases.” Both endurants and perdurants are taken to be *concrete particulars*, i.e., instances (Loux, 2002). Obviously, the notion of identity- and essence-defining properties intersect with changeability. A perdurant is typically taken to be a *spacetime worm*, i.e., an object that persists (perdures) through spacetime by way of having different temporal parts at what would be different times (temporal non-locality), but a view of

instantaneous stages is possible too (Sider, 2002). An endurant goes through time (endures), with identity/essence-defining properties that perhaps depend on occurrent objects but are not essentially constituted by those occurrent objects. The crucial distinction between these constructs is that of the nature of the identifying essential properties of the object and its change or non-change, usually defined with respect to time. Related to the distinction is the notion of *temporal parts*, i.e. whether or not a given object has temporal parts and the nature of those parts. But it is not just that distinction that defines 3D and 4D views, since some 3D perspectives permit instantaneous objects to be the temporal parts of themselves (Sider, 2002). For our purposes here, however, we will equate endurantism with the 3D view, and perdurantism with the 4D view.

A *partonomic* hierarchy, for example, is usually defined in terms of a special *partonomic* relation, the part-of relation. *Mereology* is the analysis of the part-of relation and the set of axioms that seem to constitute our notion of what a part is. In modern ontological axiomizations, mereology is combined with *topology* (connectedness among objects) to be *mereotopology* (Smith, 1996; Casati and Varzi, 1999) since *parthood* really does seem to require either point “touching”, overlap, or transitivity of those (i.e., the ‘southern edge of London’ is part of London or connected to those regions which are part of southern London). Here we begin to get into notions of granularity and vagueness, and so we’ll end our discussion (but, see: Obrst and Mani, 2000; Williamson, 1998; Keefe and Smith, 1999; Bittner and Smith, 2001).

2.3.4 Upper Ontology Initiatives and Candidates

There are a number of ongoing initiatives to define a standard upper ontology. Two initiatives that began in the early 2000s and recently ended were the IEEE Standard Upper Ontology Working Group (SUO WG)¹¹ and WonderWeb.¹² IEEE SUO WG was a standards effort operated under the IEEE Standards Association and sponsored by the IEEE Computer Society Standards Activities Board. Its goal is to specify an upper ontology that will enable computers to use it for applications such as data interoperability, information search and retrieval, automated inferencing, and natural language processing. IEEE SUO WG proposed three candidate upper ontologies, namely Suggested Upper Merged Ontology (SUMO), Upper Cyc Ontology (UCO)¹³ and Information Flow Framework (IFF).

WonderWeb was a project consortium of universities and Industry, working in cooperation with the DARPA DAML program and the W3C. WonderWeb defined a library of foundational ontologies that cover a wide range of application domains.

¹¹ IEEE Standard Upper Ontology. <http://suo.ieee.org/>.

¹² WonderWeb Website. <http://wonderweb.semanticweb.org/objectives.shtml>. Completed, July 2004.

¹³ Upper Cyc. <http://www.cyc.com/cycdoc/vocab/vocab-toc.html>.

This library is intended to be used as a basis for the development of more detailed domain ontologies. Currently three modules exist: DOLCE, OCHRE, and BFO (Masolo et al., 2003; Schneider, 2003).¹⁴

In addition, there have been proposed other upper (foundational) ontologies, including Generalized Ontological Language (GOL)/ General Formal Ontology (GFO) (Heller and Herre, 2004; Herre et al., 2006),

For comparisons of upper ontologies, see Grenon (2003); Semy et al. (2005); and Mascardi, Cordi, Rosso (2006). There was also an effort in 2006 by the Ontolog Forum called the Upper Ontology Summit,¹⁵ at which many major upper ontology developers signed a joint communiqué to agree “to develop the mechanisms and resources needed to relate existing upper ontologies to each other in order to increase the ability to reuse the knowledge to which they give access and thereby facilitate semantic interoperability among those other ontologies that are linked to them” (Obrst et al., 2006).

2.4 Structuring the Ontological and Meta-Ontological Space

This section extends the ontological architecture considerations of the previous section in two ways: (1) it moves beyond purely vertical considerations of object level ontologies (upper, middle, domain) to include structural and logical relations among the ontologies of those levels, and ways of addressing the entire object level space, which we are calling the *ontological space*, and so necessarily involving notions of modularity and context (applicability of assertions); (2) it addresses also the *meta-ontological space*, i.e., the knowledge (ontology) representation (KR) space at the meta-level to the ontology object level. Although both of these topics require a more lengthy elaboration than we can provide here, we will sketch out some of the considerations and approaches. Because the two topics are so intricately connected, we flatten the structure of our exposition somewhat, addressing meta-ontological issues and then ontological issues, acknowledging explicitly that the latter depend on the former – even when a formalized connection cannot yet be established.

2.4.1 Knowledge Representation Languages and Meta-Ontologies

Another way of viewing ontological architecture is more abstractly, i.e., meta-ontologically, in terms of the representation levels. These representation levels include minimally: (1) the knowledge (ontology) representation language level; (2) the ontology concept (universals) level; (3) the ontology instance (particulars)

¹⁴Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) Website. <http://www.loa-cnr.it/DOLCE.html>.

¹⁵Upper Ontology Summit, Ontolog Forum, 2006. <http://ontolog.cim3.net/cgi-bin/wiki.pl?UpperOntologySummit>.

level. These are displayed in Fig. 2.7. The knowledge representation (KR) level is the meta-level to the ontology content level (which is its object level). The KR level characterizes what can be expressed in the ontology. The ontology concept level is the level that characterizes the generic descriptions of the ontology, i.e., universals or categories, the ontology proper, which might be considered either the organizing structure for the ontology instance level or the intensional level which describes the properties that will hold of specific individuals (the extension) at the ontology instance level. The third level (instances or particulars) is the level that instantiates the universals expressed at the second level (universals).

This view partially returns to the Ontology Spectrum perspective, in which the expressiveness of the knowledge representation language determines the richness of the object level ontology assertions that can be made.¹⁶

A given ontology is syntactically expressed in a particular logical or knowledge representation language. Although the choice of knowledge representation language is secondary to the actual ontological content, it is still important because it determines whether in fact a given upper ontology can be utilized completely or just partially.

Typically, upper ontologies require expressiveness at the level of First Order Logic (FOL), but occasionally require more, i.e., second-order or higher. Second-order is required if the upper ontology quantifies over predicates (or relations or properties), though limited finite quantification over predicates (in the form of a list of predicates) can be supported in a first-order language, as KIF/Common Logic demonstrates.¹⁷

Furthermore, an upper ontology may require a modal extension of FOL, depending on how modalities such as necessity/possibility and potential modalities such as temporal/spatial operators are expressed in the ontology. In general, modalities (necessity, belief, obligation, time, etc.) can be expressed either in the (meta level) logic/KR language or in the (object level) ontology, but in either case, ways to assert and refer to modal assertions will differ. These differences may be important to the expressions a domain ontology wants to make.

If the logic/KR language in which a given upper ontology is encoded is less expressive than the logic/language in which a specific upper ontology is expressed, semantic information loss will result. The resulting encoding of the upper ontology will contain only a subset of the original expression of the ontology. For example, if the original upper ontology is expressed in KIF/Common Logic and then encoded in OWL (Bechhofer et al., 2004) only a portion will be retained in OWL, which, being a description logic-based ontology language, tries to maximize machine reasoning tractability by minimally, but definitely, limiting expressivity. OWL Full, the most expressive “dialect” of OWL, may in fact be nearly equivalent in expressivity to FOL, but remains ultimately less expressive.

¹⁶Portions of this section are adapted from Semy, Pulvermacher, Obrst (2005), pp. 5-13.

¹⁷Common Logic. <http://cl.tamu.edu/>.

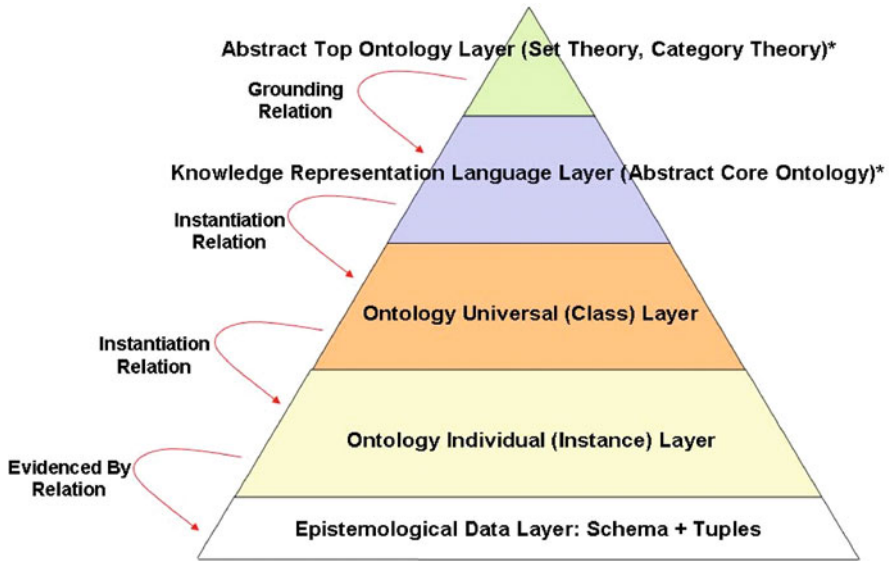


Fig. 2.8 Ontological architecture: a bigger picture

Finally, it should be noted that if KR languages are either not sufficiently formalized so that there is a clear notion of the formal semantics of the language, or are sufficiently formalized, but offer only indirect expression of upper ontology axioms, then portions of the upper ontology cannot be used by interpreting software. Portions of the upper ontology must then be annotated and interpreted solely by human beings.

A refinement of this three-level view is the (meta-)ontology architecture of Fig. 2.8.¹⁸

In this diagram, the KR language (or Abstract Core Ontology [ACO], in the usage of Herre and Loeb (2005)) is in a grounding relation to the abstract top ontology (ATO) layer, which is rooted in fundamental mathematical theories, a meta-level to our KR level; a meta-meta-level to our ontology concept/universal level, which after all is our primary interest here, i.e., it is rooted in set theory and category theory. Logic and in particular First-Order Logic presumably make up the KR/ACO level. In this view the ACO assumes at least some of the role of foundational ontologies (typically upper ontologies). For example, Herre and Loeb (2005, p. 1404) describe the basic constructs of the ACO as the following (Table 2.3), with the presumed two underlying core distinctions of the real world being that of *category* and *individual*:

¹⁸Adapted from Herre and Loeb (2005).

Table 2.3 Basic entity types and relations (Herre and Loebe, 2005)

Meta-Level Entity Types (Sets of urelements)			
<i>Name</i>	<i>Symbol</i>	<i>Name</i>	<i>Symbol</i>
Category	Cat	Individual	Ind
Object Category	OCat	Object	Obj
Property	P	Attribute	Att
Role Category	RCat	Role	Rol
Relation	R	Relator	Rel
Meta-Level Relations (Sets of pairs of urelements)			
<i>Name</i>	<i>Symbol</i>	<i>Argument Restrictions</i>	
identity	$x = y$	–	
instantiation	$x :: y$	Cat(y)	
inherence	Inh(x, y)	Att(x) or Rol(x)	
role-of role	(x, y)	Rol(x), Rel(y)	
categoryal part-of	catp(x, y)	Cat(x), Cat(y)	

The potential value of this revised architecture is that it generalizes the constructs expressible at the KR language level, on the one hand, thus enabling many kinds of KR languages to be developed and be compared, while enforcing a logical consistency on the object ontologies developed in the Ontology Universal and Ontology Particular levels. On the other hand, an ACO is grounded in the firm mathematics of the ATO, i.e., its constructs are defined in terms of set theory and category theory, and presumably some variant of formal logic.

In Herre and Loeb (2005), two different ACOs are developed as an attempt to address the requirements for an ACO: initially, CPO, which is based on categories and properties only and thus not all of the constructs in Table 2.3; and then secondly, CRO, which addresses all of the constructs in Table 2.3, including in particular, relations. Of course, both of these ACO meta-ontologies have fragments which will have some constructs but not others. The notion of *concept lattice* from formal concept analysis (Ganter and Willie, 1996) is modeled as a experiment to gauge the expressiveness of CPO in Herre and Loeb (2005). They conclude that CPO does not appear to be expressive enough for all the examples given in Ganter and Willie (1996). However, they emphasize that this formalization does highlight distinct interpretations that can exist for object ontologies, based on the use of ACOs; this result in itself is valuable and argues for a meta-ontological architecture such as they describe.

Since Herre and Lobeb (2005) axiomatize CPO and CRO with a type-free FOL, presumably FOL and other logics constitute at least the lower levels of ATO, in addition to set theory and category theory at the higher levels. So ATO is best characterized as the logico-mathematical fundamental level in this architecture.

Given that ACOs partially constitute the KR language level, they really act as an upper meta-ontological level of the meta-logical KR language level identified in Fig. 2.7. The KR languages below them presumably act partially as ACOs (I am here thinking, for example, of the implicit class theories embedded in OWL and other KR languages), i.e., as *instantiations* of an apparently unexpressed ACO.

2.4.2 The Lattice of Theories

Because ontologies are often considered theories, i.e., as from our discussion previously of strong ontologies as logical theories, then a sensible question is: what are the relationships among the theories? Intuitively these relationships are mathematical or logical in nature. Among others, John Sowa (2005) has characterized the entire structure of ontologies *qua* theories as a *lattice of theories*. Sowa states that Lindenbaum showed that the partial ordering defining the lattice of theories (Davey and Priestley, 1996) can be view in three ways, as: (1) *implication*, (2) *provability*, and (3) *semantic entailment*. These notions are derived from the Lindebaum’s infinite lattice of theories.

Figure 2.9 depicts implicitly the lattice of theories, but also a portion of the structural relationships among so-called *microtheories* and *little theories*, which are described in Section 2.4.4. This is a notional figure in which the alphabetic symbols A, B, . . . , Z (all with implicit subscripts to distinguish multiple occurrences) represent *propositions*, the symbol ‘;’ represents *disjunction*, the symbol ‘,’ represents *conjunction*, and the symbols $T_n, T_{n+1}, \dots, T_{n+i}$ ($n, i \in \text{Integers}$), along with ‘Top Theory’, represent distinct *theories*.

In this figure, therefore, T_1, \dots, T_6 represent distinct, more specific theories in the lattice of theories having as Top (most general node) the Top Theory. Top Theory

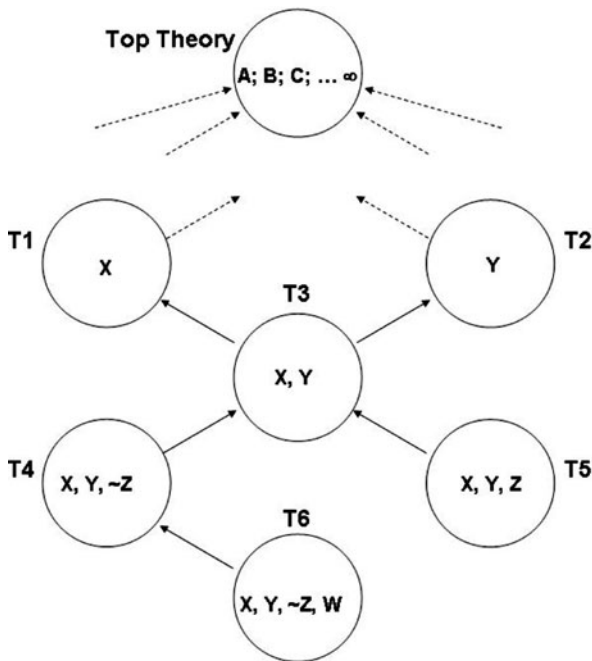


Fig. 2.9 Lattice of theories, microtheories, little theories

represents the disjunction of all possible propositions, both positive and negative (i.e., negated positive propositions).¹⁹ Arrow lines represent the lattice relations among theories, interpreted as either implication, provability, or semantic entailment (though semantic entailment might be the most perspicuous here). Dotted arrow-lines represent (infinitely) other many theories in the lattice (including, it should be mentioned, other theories below T1 – T6).

So in this view, although T4 and T5 are mutually inconsistent (because they assert, respectively, $\sim Z$ and Z), taken individually they are locally consistent. Furthermore, they are consistent with respect to T3. T3 in turn is consistent with both T1 and T2. T6 is consistent with T4 but not T5. All are consistent with Top Theory. Microtheories and little theories, as we will see, inhere in this framework. T1 – T6 can be considered microtheories or little theories.

2.4.3 Modularity and Context in the Ontological Space

The two notions of *modularity* and *context* are closely linked when we consider the larger *ontological space*. The ontological space we will define to be: the object level space of ontologies and their knowledge bases, i.e., the universals (classes, categories) and particulars (instances, individuals), and the theories and interpretations which make up this space. This section is concerned with ways that have been proposed to structure the relationships among those *modules* (theories and their interpretations).

In the 1980s, even prior to the rise of ontological engineering as a technical discipline of computer science, John McCarthy, along with his students, began to investigate the nature of context in knowledge representation and to formalize it (McCarthy, 1987, 1990, 1993; Guha, 1991; Buvač, 1993; Buvač, Buvač, Mason, 1995; Buvač 1996a-b; McCarthy and Buvač, 1997). Others took up the thread and implementations appeared, such as that of a *microtheory* in Cyc (Blair et al., 1992; Lenat and Guha, 1990).

Most formalizations of context use a specialized context lifting predicate $ist(P, C)$ or alternatively $ist(C, F)$, which means: proposition P is true in the Context C; formula F is true in Context C.²⁰ Typically contexts are first-class objects, and thus *ist* formulas are first-class formulas (Guha, 1991, p. 17). In Cyc, these contexts are implemented as *microtheories*, i.e., theories/ontologies in the general Cyc ontological space which are locally consistent, but taken together, are not globally consistent in the ontological space. In principle, this is similar to the *lattice of theories* notion discussed earlier, and also to *possible world semantics* (assuming the universes of discourse are the same) (Obrst et al., 1999), with the understanding that

¹⁹So, the theories are propositional theories here, for purposes of simplicity, but they should be understood as being formed from FOL or higher-order logical formulae, with the additional understanding that fully saturated (with instantiated, non-variable, i.e., *ground* terms) FOL or higher-order formulae are propositions.

²⁰We use propositions here, but the general case is formulae.

if a microtheory was replaced with a set of consistent theories, the node in the theory space could be viewed as the possible world in which all those theories are true, and thence the relations among those theory-worlds could be construed as *accessibility relations* as among possible worlds, with those accessibility relations macroscopically thus corresponding to the Lindenbaum view microscopically, e.g., as semantic entailment.

Menzel (1999) points out that these formalizations of context propose a so-called “subjective conception” of context, i.e., one which defines contexts as sets of propositions, as theories related via an entailment relation – so typically as a set of beliefs of a person or agent, and thus *subjective*. Contrarily, Menzel (1999) argues for an “objective conception” of context, which views the truth of a proposition not as a logical relation (such as entailment) between the proposition of a context and other propositions, but instead as a correspondence relation between the proposition and the world, and thus *objective* (Obrst and Nichols, 2005, p. 2). Menzel (1999) therefore argues for the use of *situation theory* (Barwise and Perry, 1983), which explicitly establishes more granular formal contexts in natural language semantics than the usual notion of possible worlds, i.e., situations.

Giunchiglia and Ghidini (1998), Giunchiglia (1997), and Giunchiglia and Bouquet (1998) analyze context as *Local Model Semantics*. Their formalization of context is based on two properties identified as *Locality* and *Compatibility*. Locality, in their view, is a property shared by the language, the notion of satisfiability of a formula within a specific context, and the structure of individual contexts: everything in the determination of context is local. Compatibility, they characterize as the notion of mutual influence that contexts have on themselves, including the structural notion of changing the set of local models of two contexts so that they agree to some extent. LMS defines a special *model* for two languages (L_1, L_2) which is a *compatibility relation* $C \subseteq 2^{\overline{M}_1} \times 2^{\overline{M}_2}$ (Giunchiglia and Ghidini, 1998, p. 284). Given the two languages, they associate each of them with a set $M_i \subseteq \overline{M}_i$ of local models, where \overline{M}_i is the class of all models of L_i . A specific *context* then is a set of *local models* $m \in \overline{M}_i$ allowed by C . A formula φ of a context is satisfied in model C iff it is satisfied by all the local models of C .²¹ Bouquet et al. (2003) define a context extension of the Semantic Web ontology language OWL called Context OWL (C-OWL), which is based on the formalization of LMS.

Some KR languages have been proposed, which reify contexts. In the Interoperable Knowledge Representation for Intelligence Systems (IKRIS) project,²² for example, a KR language was developed called IKL (IKRIS Knowledge Language), which can be considered an extended Common Logic, or “Common Logic on steroids” (Hayes and Menzel, 2006; Hayes, 2006). In IKL, contexts are formalized as first-class objects (Makarios, 2006a-c). But the decision was made to contextualize constants rather than sentences, and so (Welty, 2006):

²¹For further discussion of LMS, see Obrst et al. (1999) on which this current discussion is based.

²²Interoperable Knowledge Representation for Intelligence Support (IKRIS). 2006. http://nrcc.mitre.org/NRRC/Docs_Data/ikris.

Fred in (ist C0 (P Fred)) is interpreted with respect to C0

And each constant is replaced with a function of the context and the constant:

$$\{(\text{forall}(x) (\text{implies} (\text{P} (\text{iso CMX})) (\text{G} (\text{iso CMx})))));$$

$$(\text{P}(\text{isoC0Fred}))\}$$

Some questions to ask about contexts with respect to ontologies are the following. What is the relationship between a context and an ontology? What is the relationship between a context and a module of an ontology? Are contexts and ontologies distinct or the same? Is a context embedded within a given ontology (where the ontology is viewed as a theory or set of logical theories about a domain)? Is a context extraneous to an ontology and thus outside the ontology as theory, leading us to view a context as encapsulating ontologies and changing the interpretations of those ontologies in this context as opposed to that context (Obrst and Nichols, 2005)? In this section, we will discuss modularization in ontologies, and we will treat *contexts* as being essentially about perspectives (akin to but more complex than *views* in the relational database world), i.e., as logical theories (and their interpretations), which in our estimation are what ontologies are.

Since 1997, formalization of context has established itself as a technical thread in the CONTEXT conferences (CONTEXT 97, Brézillon and Cavalconti, 1997; CONTEXT 07, Kokinov et al., 2007). Modularity of ontologies in its own right has been addressed by very recent workshops [Haas et al., 2006; Cuenca-Grau et al., 2007]. It is often remarked on that formalized context and ontology modules bear a close resemblance and depend on each other, which has led to the recent Context and Ontologies Workshops (Bouquet et al., 2007), and see in particular (Loeb, 2006; Bao and Honavar, 2006; Lüttich et al., 2006; Kutz and Mossakowski, 2007).

2.4.4 Microtheories, Little Theories, Ontology Versioning

A microtheory is a theory in Cyc (Blair et al., 1992; Kahlert and Sullivan, 2006) which is a portion of the (monolithic) ontology that is separable from other microtheories, and thus with respect to those possibly containing contradictory assertions. A Cyc microtheory “is essentially a bundle of assertions that share a common set of assumptions; some microtheories are focused on a particular domain of knowledge, a particular level of detail, a particular interval in time, etc. The microtheory mechanism allows Cyc to independently maintain assertions which are *prima facie* contradictory, and enhances the performance of the Cyc system by focusing the inferencing process.”²³

See Fig. 2.9, previously introduced in the discussion of the lattice of theories. Microtheories represent an implementation of a formalization of context

²³What’s in Cyc? http://www.cyc.com/cyc/technology/whatis_cyc_dir/whatsincyc.

deriving from McCarthy (1987, 1990, 1993), but focused in particular on Guha (1991). Originally Cyc microtheory contexts consisted of two parts: assumptions and content. Subsequently, due to computational inefficiencies with the formalism (primarily in the cost of so many *liftings* of assertions from one context into another), the microtheory was recast with finer internal structure. Lenat (1998) identified 12 dimensions of context space along which contexts vary in Cyc: “Absolute Time, Type of Time, Absolute Place, Type of Place, Culture, Sophistication/Security, Granularity, Epistemology, Argument-Preference, Topic, Justification, and Anthropacity” (Lenat, 1998, p. 4), with each primary dimension being itself a bundle of partially mutually dependent finer-grained dimensions. A richer calculus of contexts is thus required. In our view, a dimension of context-space is thus similar to an index of common and commonsense world knowledge which cross-cuts domain theories (Obrst et al., 1999, p. 6). This latter usage more closely corresponds to Lewis’s (1980) ‘index’, as opposed to his much richer ‘context’ (and see discussion in Giunchiglia and Bouquet (1998), p. 7).

In Farmer et al. (1992) and Farmer (1996, 2000), a formalization and implementation (for mathematical theorem proving) of a notion similar to that of microtheories is introduced, that of *little theories*. Defining a theory as a set of axioms in a formal language, Farmer et al. (1992) contrast two predominant views in mathematics: (1) the *big theory* approach, which is one large theory of very expressive axioms (such as Zermelo-Fraenkel set theory) such that the models of these axioms will contain all of the objects of potential interest to the mathematician; (2) the *little theory* approach, in which a number of theories will be used and different theorems are proven in different theories, depending on the amount of structure needed. The little theory approach uses *theory interpretations* as the primary formal notion, where theory interpretation is defined as “a syntactic translation between the languages of two theories which preserves theorems” (Farmer et al., 1992, p. 2). A formula which is a theorem in the source theory is thus translated into a theorem in the target theory, which requires the source theory axioms to be translated into theorems in the target theory. Theorems can thus be reused in different theories, and one can establish the consistency of the source theory with respect to the target theory. A theory interpretation between two theories also enables inferring a relation between the models of the two theories. Farmer et al. (1992) and Farmer (1996, 2000) also establish how such a formalization can be used to implement a proof system for doing semi-automated mathematical proofs, developing the Interactive Mathematical Proof System (IMPS). IMPS enables one to store theories and theory interpretations. The little theory approach therefore allows for a network of theories and provides both *intertheory* and *intratheory* reasoning. So, similar to the notion of microtheories, little theories enable different perspectives (different contexts) to be represented as different theories (microtheories), with the formal device enabling switching from one theory to another being the notion of a theory interpretation between the two theories, preserving theoremhood between the two theories.

Finally, De Leenheer (2004), De Leenheer and Mens (2007), De Leenheer et al. (2007) demonstrates the significance of the relation among ontologies, contexts, and

versions of ontologies (among other perspectives), by introducing a formal framework for supporting context driven ontology engineering based on the DOGMA²⁴ framework and methodology. Note that this is not a formalization, but instead an elaboration and use of existing notions from context formalization.

2.4.5 Information Flow Framework Meta-Ontology

The Information Flow Framework (IFF)²⁵ is authored by Bob Kent (Kent, 2004, 2006) and was recently being developed under the IEEE SUO Working Group.²⁶ IFF provides a framework for sharing ontologies, manipulating ontologies as objects, relating ontologies through morphisms, partitioning ontologies, composing ontologies via fusions, noting dependencies between ontologies, and declaring the use of other ontologies. It takes the building block approach to ontology construction and management, using category theory (Mac Lane, 1971; Barr and Wells, 1999) and Information Flow Theory (IFT) (Barwise and Seligman, 1997) to support ontology modularity.

IFT is a framework more general than possible worlds, allowing also *impossible worlds* (Barwise, 1998). Figure 2.10, for example, displays the interpretation of one language or logic L_1 (*classification* in their terminology) into another L_2 , with the accompanying association with every structure M_2 for the logic L_2 a structure M_1 for the logic L_1 . One also has to make sure that $M_1 \models_{L_1} \alpha^1$ iff $M_2 \models_{L_2} \alpha^2$ holds for all structures M_2 for L_2 and all sentences α of L_1 Barwise and Seligman (1997, p. 32; Obrst et al., 1999, p. 7).

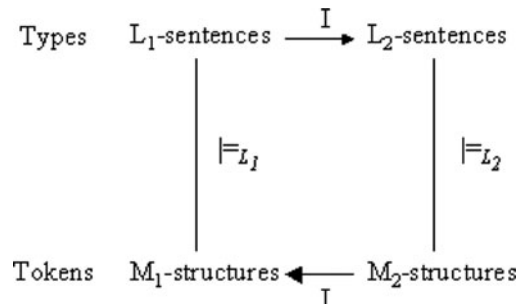


Fig. 2.10 Interpretation of languages

²⁴Developing Ontology-Grounded Methods and Applications (DOGMA); a research initiative of the Free University of Brussels, Semantic Technologies and Applications Lab (VUB STARLab). <http://www.starlab.vub.ac.be/website/>.

²⁵Information Flow Framework. <http://suo.ieee.org/IFF/>. See also: <http://www.ontologos.org/IFF/IFF.html>.

²⁶IEEE Standard Upper Ontology. <http://suo.ieee.org/>.

IFF is primarily a meta-ontology, to be used for ontology-ontology integration, and is still in early stages of development. The original intent for the IFF was to define morphisms for concept lattices, mapping between Formal Concept Analysis (Ganter and Willie, 1996) and Information Flow classifications. In addition, the IFF attempts to develop a framework using IFT and category theory to implement the so-called “lattice of theories” view of the linkages among ontologies at the object level (Kent, 2003). The IFF is a fairly detailed framework, and within the limited space of this paper, no thorough elaboration can be given here. The interested reader is instead invited to peruse the IFF web site.²⁷ The primary architecture of the IFF can be depicted as in Fig. 2.11, which is from Kent (2002). See also Kalfoglou and Schorlemmer (2002, 2003, 2004) for discussion of IFF and issues in ontology mapping.

In the IFF, object level ontologies (upper, mid-level, domain) reside at the object level, and their constructs are defined at the meta-level in a series of ascending sub-levels, from IFF ontology (axiomatization) lower level to higher components. The lower meta-level defines, axiomatizes, and reasons about *particular* categories, functors, adjunctions, colimits, monads, classifications, concept lattices, etc., whereas the upper meta-level defines, axiomatizes, and reasons about *generic* categories, functors, adjunctions, colimits, monads, classifications, concept lattices, etc. The top meta-level was a formalization of the Knowledge Interchange Format (KIF).²⁸

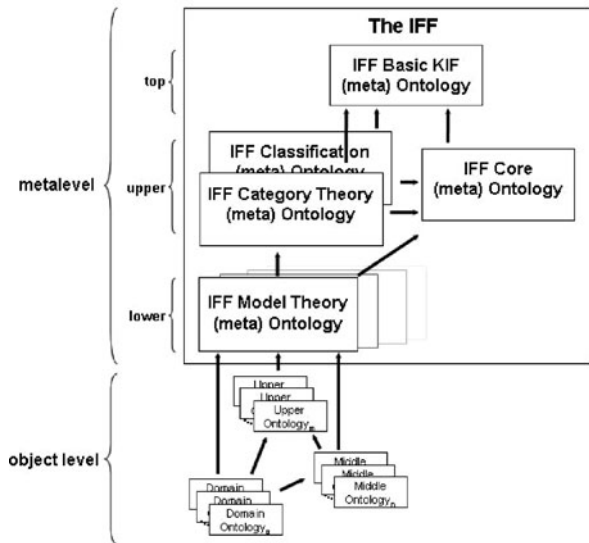
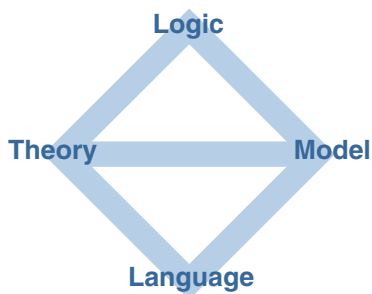


Fig. 2.11 Information flow framework: architecture (2002 version, slide 9)

²⁷<http://www.ontologos.org/IFF/IFF.html>.

²⁸Originally, Knowledge Interchange Format (KIF) Specification (draft proposed American National Standard [dpANS] NCITS.T2/98-004: <http://logic.stanford.edu/kif/dpans.html>. However,

Fig. 2.12 Objects of IFF
(Kent, 2005, slide 30)



In Fig. 2.12 is a picture of the primary objects in IFF: Logics, Theories (syntactic representations), Models (semantic representations), and Languages.

IFF builds on IFT, which in turn builds on category theory. But a formalization which is very close to IFF is that of *institutions* of Goguen (1991), Goguen and Burstal (1992), Goguen and Roşu (2002), and the related work of Kutz and Mossakowski (2007), and Schorlemmer and Kalfoglou (2008). *Institutions* formalize the notion of a logical system by demonstrating that there is a satisfaction relation between models and sentences which is consistent under change of notation (Goguen and Burstal, 1992; Goguen 2006). *Institutions* formalize category-theoretic *signatures* derived from (generalized from) vocabularies (ontologies) and *signature morphisms* derived from (generalized from) mappings between vocabularies (ontologies). So institutions are “truth preserving translations from one logical system to another”, which is very similar to the intention of the IFF.

It should also be noted that the IFF, though a meta-level architecture, is more of a meta-logical architecture, rather than a meta-ontological architecture as Herre and Loebe (2005, p. 1411) point out, so – using the latter’s terminology – the IFF is more like a very elaborated ATO rather than an ACO, in the terminology of Herre and Loebe (2005). However, given that ACOs act as the upper level of the KR language meta-logical level (the upper level in Fig. 2.7) (and so, specific KR languages can be seen as both *languages* and partial instantiations of implicit ACOs, i.e., as embodying meta-ontological *theories*), it does seem that the logical vs. ontological levels of the meta-level needs to be better spelled out. This discussion also demonstrates that further analysis of the interplay between the logical and ontological levels is a fruitful subject of study.

For further information about the IFF, the interested reader is referred to Kent (2010, this volume).

KIF has been superseded by Common Logic, which includes a KIF-like instantiation called CLIF, and is now an ISO standard.

2.5 What the Future Holds: A Vision

In this chapter, we have looked at ontological architecture, what it consists of and what it is related to. We began our discussion by delineating ontological architecture from ontology architecture, and observed that we would necessarily focus on the former – as being the necessary foundation for the latter. As one can see, there is clear technical apparatus emerging that addresses the logical, meta-ontological, and ontological architectures and their requirements to support the actual development and deployment of ontologies as engineering products in an ontology architecture – which itself encompasses ontology lifecycle management as a practical discipline wedding ontologies and software development.

Along the way, we discussed not just the components of ontological architecture, but necessarily aspects of logical and meta-ontological architecture. We tried to relate these three notions systematically and consistently in a larger framework that we hope will provide support for subsequent ontology architecture. There are many moving pieces to this architectural puzzle. Ontology engineering as a branch of both computer science and software engineering has just recently emerged – and is propelled by ideas from formal ontology in philosophy, formal logic in mathematics, formal semantics in linguistics, formal methods and applications in computer science and artificial intelligence, and formal theories in cognitive science. There

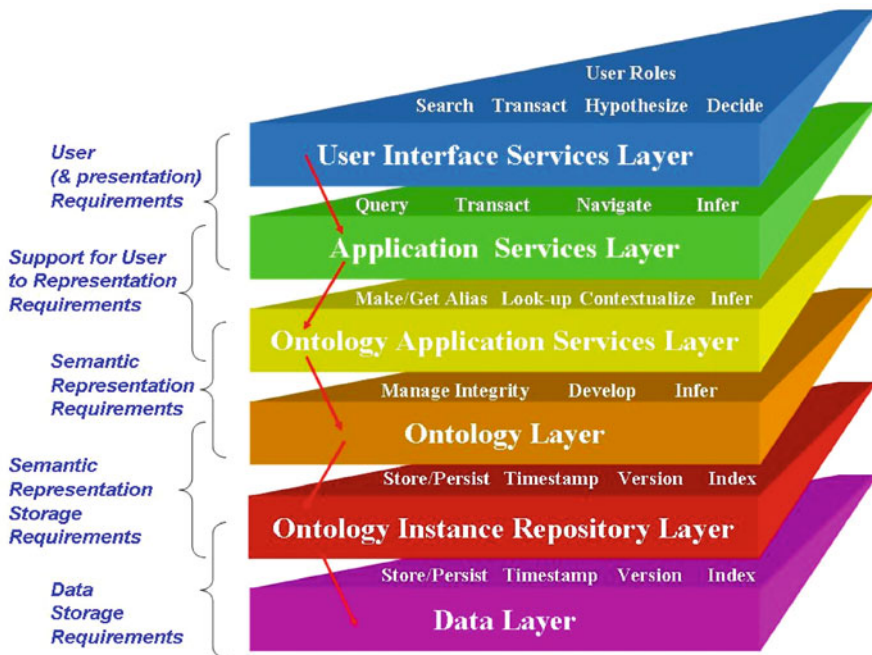


Fig. 2.13 Ontology architecture: application layers

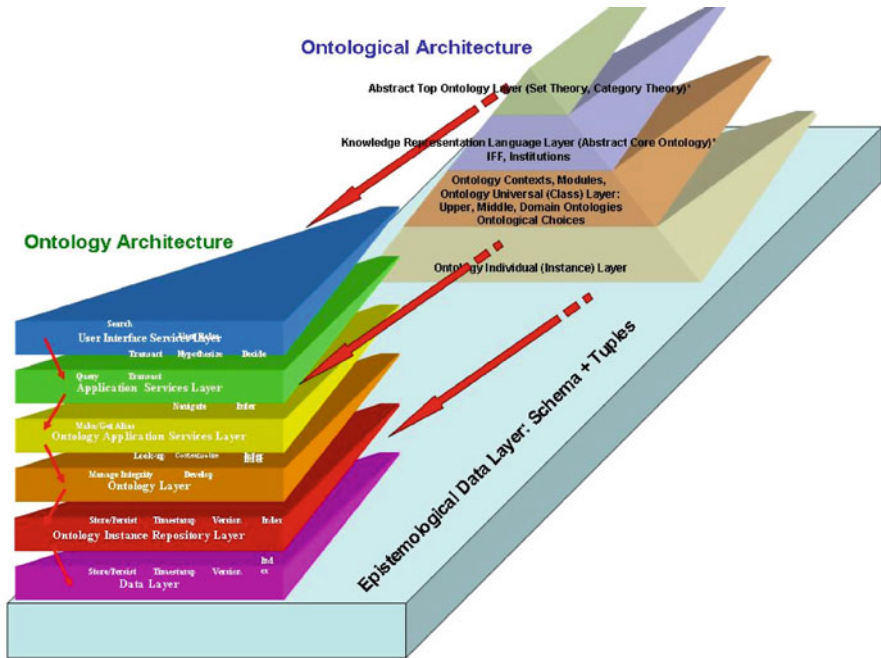


Fig. 2.14 Ontological architecture supporting ontology architecture

is a grand fermenting of philosophical, logico-mathematical, scientific, and engineering ideas that make the future uncertain to predict. There are, however, some indications, enough for a vision for the future.

Figure 2.13 depicts one view of an *ontology architecture*, which admittedly we could not address in this chapter. However, this architecture represents a sound view of what is architecturally necessary to deploy ontologies in the real world. Each of these layers are significant and necessary for an application based on ontologies. Each layer constitutes hefty portions of a distinct chapter on actually using engineered ontologies to assist users by way of their software applications and services in the near to foreseeable future.

Behind this figure and its depicted architecture, however, is another figure and its depicted architecture. Figure 2.14 shows the notional relationship between the ontological architecture and the ontology architecture.

This is the future: sound ontology philosophy and science driving sound ontology engineering, with many other technical disciplines collaborating to provide sound ontology-based applications.

Acknowledgments and Disclaimers The author’s affiliation with The MITRE Corporation is provided for purposes only, and is not intended to convey or imply MITRE’s concurrence with, or support for, the positions, opinions or viewpoints expressed by the authors. I note that the views expressed in this paper are those of this author alone and do not reflect the official policy or position of any other organization or individual. I acknowledge deep appreciation for reviews of this material and suggestions to improve it by Roberto Poli and anonymous reviewers. Discussions

(mostly electronic, but occasionally live) on these topics have been much appreciated, with Roberto Poli, John Sowa, Bill Andersen, members of the MITRE Information Semantics group including Ken Laskey, David Ferrell, Deborah Nichols, Mary Parmelee, Merwyn Taylor, Karla Massey, Jonathan Tivel, Frank Zhu, Richard MacMillan, and former or auxiliary members including Ken Samuel and Allen Ginsberg, and friends and colleagues including especially Suzette Stoutenburg (my best colleague and friend), Arnie Rosenthal (my best devil's advocate and good friend), Len Seligman, Marwan Sabbouh, Peter Mork, my persevering and nascent ontologist wife Christy Obrst, and a pack of good dogs, cats, and chickens to mostly keep me level.

References

- Andersen, B., R. Kohl, J. Williams. 2006. Ontology lifecycle management. Ontology Works, Inc. Version 1.0. Oct 14, 2005. ontologyworks.com.
- Bao, J., and V.G. Honavar. 2006. *Divide and conquer semantic web with modular ontologies - A brief review of modular ontology language formalisms*. Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06, CEUR Workshop Proceedings, vol. 232. eds. H. Haas, S. Kutz, and A. Taminin, 5 Nov 2006, Athens, GA.
- Barr, M., and C. Wells. 1999. *Category theory for computing science*, 3rd edn. Montreal, QC: Les publications Centre de recherches mathématiques.
- Barwise, J. 1998. Information and impossibilities. Manuscript. <http://www.phil.indiana.edu/~barwise/>
- Barwise, J., and J. Seligman. 1997. *Information flow: The logic of distributed systems*. Cambridge, UK: Cambridge University Press.
- Barwise, J., and J. Perry. 1983. *Situations and attitudes*. Cambridge, MA: MIT Press.
- Basic Formal Ontology (BFO): <http://www.ifomis.uni-saarland.de/bfo>
- Basin, D., M.D. Agostino, D. Gabbay, S. Matthews, and L. Viganò, eds. 2000. *Labelled Deduction*, 107–134, Scotland, UK: Kluwer Academic Publishers, 2000.
- Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L>A. Stein. 2004. OWL web ontology language reference. W3C recommendation. 10 Feb 2004. <http://www.w3.org/TR/owl-ref/>
- Bittner, T., and B. Smith. 2001. Granular partitions and vagueness. In *Formal ontology and information systems*, eds. C. Welty, and B. Smith, 309–321. New York, NY: ACM Press.
- Bittner, T., and B. Smith. 2003. Endurants and perdurants in directly depicting ontologies. Draft. <http://ontology.buffalo.edu/smith/articles/EPDDO.pdf>
- Blackburn, P. 1999. Internalizing labelled deduction. In Proceedings of Hylo'99, First International Workshop on Hybrid Logics. 13 July 1999, Saarbrücken, Germany. Published in 2000. *Journal of Logic and Computation* 10(1):137–168.
- Blair, P., R.V. Guha, and W. Pratt. 1992. Microtheories: An ontological engineer's guide. Technical report Cyc-050-92, 5 Mar 1992, Cycorps, Austin, TX. <http://www.cyc.com/tech-reports/cyc-050-92/cyc-050-92.html>
- Bouquet, P., J. Euzenat, C. Ghidini, D.L. McGuinness, L. Serafini, P. Shvaiko, and H. Wache, eds. 2007. Proceedings of the International Workshop on Contexts and Ontologies: Representation and Reasoning (C&O:RR). Collocated with the 6th International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-2007) Roskilde, Denmark, 21 Aug. <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-298/>
- Bouquet, P., F. Giunchiglia, F. Van Harmelen, L. Serafini, and H. Stuckenschmidt. 2003. C-OWL: Contextualizing ontologies. 2nd International Semantic Web Conference (ISWC 2003), eds. D. Fensel, K.P. Sycara, and J. Mylopoulos, 164–179. Sanibel Island, FL, 20–23 Oct 2003.
- Brézillon, P., and M. Cavalcanti. 1997. Modeling and Using Context: Report on the First International and Interdisciplinary Conference, CONTEXT-97, Rio de Janeiro, RJ, Brazil on 4–6 Feb 1997. *The Knowledge Engineering Review* 12(4):1–10.

- Buckland, M.K., and F. Gey. 1994. The relationship between recall and precision. *Journal of the American Society for Information Science* 45(1):12–19.
- Buvač, S., V. Buvač, and I.A. Mason. 1995. Metamathematics of contexts. *Fundamenta Informaticae* 23(3):149–174.
- Buvač, S. 1996. Resolving lexical ambiguity using a formal theory of context. In *Semantic ambiguity and underspecification*. CSLI Lecture Notes, Center for Studies in Language and Information, Stanford, CA, 1996.
- Buvač, S, ed. 1996. Proc AAAI-95 Fall Symposium on Formalizing Context. <http://www-formal.stanford.edu:80/buvac/95-context-symposium>
- Casati, R., and A.C. Varzi. 1999. Parts and places: The structures of spatial representation. Cambridge, MA: MIT Press.
- Cheikes, B. 2006. MITRE Support to IKRIS, Final Report. MITRE Technical Report MTR060158, 5 Dec 2006. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/MITRE_IKRIS_Final_Report_05Dec06.doc
- CMMI® for Development, Version 1.2. CMMI-DEV, V1.2. CMU/SEI-2006-TR-008. ESC-TR-2006-008. CMMI Product Team, Aug 2006. <http://www.sei.cmu.edu/cmmi/models/model-v12-components-word.html>
- Colomb, R.M. 2002. Use of Upper Ontologies for Interoperation of Information Systems: A Tutorial, Technical Report 20/02 ISIB-CNR, Padova, Italy, Nov 2002.
- Common Logic. <http://cl.tamu.edu/>
- Cuenca-Grau, B., V. Honavar, and A. Schlicht, F. Wolter. 2007. Second International Workshop on Modular Ontologies. 28 Oct 2007. Whistler, BC. <http://webum.uni-mannheim.de/math/lski/WoMO07/>
- Cycorp, Inc.: The CYC Technology. <http://www.cyc.com/tech.html>
- Daconta, M., L. Obrst, and K. Smith. 2003. *The semantic web: The future of XML, web services, and knowledge management*. New York, NY: John Wiley, June 2003.
- Davey, B.A., and H.A. Priestley. 1991. *Introduction to lattices and order*. Cambridge, UK: Cambridge University Press.
- Degen, W., B. Heller, H. Herre, and B. Smith. 2001. GOL: A general ontological language. In Guarino, Welty, Smith, 34–46. FOIS 2001.
- De Leenheer, P. 2004. Revising and Managing Multiple Ontology Versions in a Possible Worlds Setting. In Proc. of On The Move to Meaningful Internet Systems Ph.D. Symposium (OTM 2004) (Agia Napa, Cyprus), LNCS 3292, 798–818. Collega Park, MD: Springer.
- De Leenheer, P. and T. Mens. (2007). Ontology evolution: state of the art and future directions. In *Ontology management for the semantic web, semantic web services, and business applications, from semantic web and beyond: Computing for human experience*, eds. M. Hepp, P. De Leenheer, A. de Moor, and Y. Sure. Heidelberg, Springer.
- De Leenheer, P., A. de Moor, and R. Meersman. (2007). Context dependency management in ontology engineering: A formal approach. *Journal on data semantics VIII*, LNCS 4380, 26–56. New York, NY: Springer.
- Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) Website. <http://www.loa-cnr.it/DOLCE.html>
- Dumontier, M., and N. Villanueva-Rosales. 2007. Three-layer OWL ontology design. Second International Workshop on Modular Ontologies. 28 Oct 2007. Whistler, BC.
- Euzenat, J., A. Zimmermann, and F. Freitas. 2007. Alignment-based modules for encapsulating ontologies. Second International Workshop on Modular Ontologies. 28 Oct 2007. Whistler, BC.
- Farmer, W.M. 2000. An infrastructure for intertheory reasoning. In *Automated Deduction--CADE-17*, ed. D. McAllester, 115–131. LNCS, vol. 1831. <http://imps.mcmaster.ca/doc/intertheory.pdf>
- Farmer, W.M. 1996. Perspective switching using theories and interpretations. In *Intelligent Systems: A Semiotic Perspective*, eds. J. Albus, A. Meystel, and R. Quintero, 206–207. Vol. I, Gaithersburg, MD: National Institute of Standards and Technology, 20–23 Oct 1996. Abstract.

- Farmer, W.M., J.D. Guttman, and F.J. Thayer. 1992. Little theories. In *Automated Deduction--CADE-11*, LNCS, vol. 607, ed. D. Kapur, 567–581. <http://imps.mcmaster.ca/doc/major-imps-papers.html>
- Fernández, M. 1999. Overview of methodologies for building ontologies. Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. (IJCAI99). Aug 1999. <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/4-fernandez.pdf>
- Fernández, M., A. Gómez-Pérez, and N. Juristo. 1997. METHONTOLOGY: From ontological art to ontological engineering. Workshop on ontological engineering. AAAI Spring Symposium Series. AAAI-97, Stanford University.
- Fikes, R., C. Welty. 2006. Interoperable knowledge representation for intelligence support (IKRIS). Advanced research and development activity (ARDA)/disruptive technology office (DTO). Final briefing, Nov 2006.
- Gabbay, D. 1996. *Labelled deductive systems, principles and applications. Vol 1: Introduction*. Oxford: Oxford University Press.
- Ganter, B., and R. Wille. 1996. *Formal concept analysis: Mathematical foundations*. Berlin, Heidelberg, New York, NY: Springer.
- Gašević, D., D. Djurić, and V. Devedžić. 2006. *Model driven architecture and ontology development*. Berlin Heidelberg, New York, NY: Springer.
- Giunchiglia, F., P. Bouquet. 1997. Introduction to contextual reasoning: An artificial intelligence perspective. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9705–19, May 1997.
- Giunchiglia, F., and P. Bouquet. 1998. A Context-Based Framework for Mental Representation. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9807-02, July 1998.
- Giunchiglia, F., and C. Ghidini. 1998. Local models semantics, or contextual reasoning = locality + compatibility. In *Principles of knowledge representation and reasoning (KR'98)*, eds. A. Cohn, L. Schubert, and S. Shapiro, 282–289. Proceedings of the Sixth International Conference, Trento, Italy, 2–5 June 1998.
- Goguen, J., and G. Rosu. 2002. Institution morphisms. *Formal Aspects Of Computing* 13:274–307.
- Goguen, J.A. 1991. A categorical manifesto. *Mathematical Structures in Computer Science* 1:49–67.
- Goguen, J.A., and R.M. Burstall. M. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM* 39:95–146.
- Goguen, J. 2006a. Institutions. <http://www-cse.ucsd.edu/~goguen/projs/inst.html>
- Goguen, J. 2006b. Information integration in institutions. *For jon barwise memorial volume*, ed. L. Moss, Indiana: Indiana University Press.
- Guarino, N, ed. 1998. *In Formal ontology in information systems*. Amsterdam: IOS Press. Proceedings of the First International Conference (FOIS'98), 6–8 June, Trento, Italy.
- Guarino, N., C. Welty, B. Smith, eds. 2001. *The Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-01)*, Oct 16–19 2001, Ogunquit, Maine. ACM Press Book Series, Sheridan Publishing, Inc. <http://www.fois.org/fois-2001/index.html>
- Guha, R.V. 1991. Contexts: A formalization and some applications. PhD thesis, Stanford University. Also published as technical report STAN-CS-91-1399-Thesis, and MCC Technical Report Number ACT-CYC-423-91.
- Haase, P., V. Honavar, O. Kutz, Y. Sure, and A. Tamilin, eds. 2006. Proceedings of the 1st international workshop on modular ontologies. WoMO'06. 5 Nov 2006, Athens, Georgia, USA. CEUR Workshop Proceedings, vol. 232. <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-232/>
- Hamlyn, D.W. 1984. *Metaphysics*. Cambridge: Cambridge University Press.
- Hayes, P.J. 2006. IKL Guide. On behalf of the IKRIS Interoperability Group. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/ICL_guide.pdf
- Hayes, P.J., and C. Menzel. 2006. IKL Specification Document. Unpublished IKRIS memorandum. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/ICL_spec.pdf

- Hayes, P., F. Lehmann, C. Welty. 2002. Endurantism and Perdurantism: An Ongoing Debate. IEEE SUO email exchange, compiled and edited by Adam Pease. <http://www.ontologyportal.org/pubs/dialog-3d-4d.html>
- Heller, B., H. Herre. 2004. Ontological Categories in GOL [Generalized Ontological Language]. *Axiomathes* 14:57–76.
- Herre, H., and F. Loebe. 2005. A Meta-ontological architecture for foundational ontologies. In *CoopIS/DOA/ODBASE 2005*, LNCS 3761, ed. R. Meersman, and Z. Tari, 1398–1415. Berlin Heidelberg: Springer.
- Herre, H., B. Heller, P. Burek, R. Hoehndorf, F. Loebe, and H. Michalek. 2006. General formal ontology (GFO) – part I: Basic principles. *Technical report 8*, Germany: Onto-Med, University of Leipzig.
- Higginbotham, J., F. Pianesi, and A. Varzi, eds. 2000. *Speaking of events*. Oxford Oxford University Press.
- Hobbs, J.R., and F. Pan, 2006. Time Ontology in OWL. W3C Working Draft 27 Sept 2006. <http://www.w3.org/TR/owl-time/>
- Hodges, W. 1997. *A shorter model theory*. Cambridge, UK: Cambridge University Press. Reprinted in 2000.
- IEEE Standard Upper Ontology (SUO). <http://suo.ieee.org/>
- Interoperable Knowledge Representation for Intelligence Support (IKRIS). 2006. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/
- Information Flow Framework. <http://suo.ieee.org/IFF/> See also: <http://www.ontologos.org/IFF/IFF.html>
- Izza, S., L. Vincent, and P. Burlat. 2005. Ontology urbanization for semantic integration: Dealing with semantics within large and dynamic enterprises. Proceedings of the 2005 Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05).
- Kahlert, R.C., and J. Sullivan. 2006. Microtheories. In *First international workshop: Ontology based modeling in the humanities*, eds. W. von Hahn, and C. Vertan, 7–8 Apr 2006, University of Hamburg (Bericht 264).
- Kalfoglou, Y., and M. Schorlemmer. 2004. In *Ontology mapping: The state of the art*, eds. A. Sheth, S. Staab, and M. Uschold, Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings on Semantic Interoperability and Integration Internationales Begegnungs- und Forschungszentrum (IBFI), <http://drops.dagstuhl.de/opus/volltexte/2005/40/pdf/04391.KalfoglouYannis.Paper.40.pdf>
- Kalfoglou, Y., and M. Schorlemmer. 2003. Ontology mapping: The state of the art. *Knowledge Engineering Review* 18(1):1–31.
- Kalfoglou, Y., and M. Schorlemmer. Information-flow-based ontology mapping. In *On the move to meaningful internet systems 2002: CoopIS, DOA, and ODBASE*. Lecture Notes in Computer Science 2519, 1132–1151. Heidelberg: Springer.
- Keefe, R., and P. Smith, eds. 1999. *Vagueness: A reader*. Cambridge, MA: MIT Press.
- Kent, R.E. 2002. The IFF approach to semantic integration. Power point presentation at the boeing Mini-workshop on semantic integration, 7 Nov 2002. <http://www.ontologos.org/Papers/Boeing%20Mini-Workshop.ppt>
- Kent, R.E. 2003. The IFF approach to the lattice of theories. Unpublished manuscript, Apr 22 2003. Formerly at: <http://suo.ieee.org/IFF/lattice-of-theories.pdf>
- Kent, R.E. 2004. The IFF foundation for ontological knowledge organization. In *Knowledge organization and classification in international information retrieval*, eds. N.J. Williamson, and C. Beghtol, 187–203. Volume 37 of Cataloging & Classification Quarterly. New York: Haworth Press.
- Kent, R.E. 2006. The information flow framework: New architecture. International Category Theory Conference (CT 2006), White Point, Nova Scotia, 25 June – 1 July 2006.
- Kent, R.E. 2010. The information flow framework. Chapter in Part One: Ontology as Technology in the book: *TAO – Theory and applications of ontology, volume 2: The information-science stance*, eds. M. Healy, A. Kameas, and R. Poli.

- Kiryakov, A., K. Simov, and M. Dimitrov. 2001. Onto map: Portal for upper-level ontologies. In *Proceedings of the International Conference on Formal Ontology in Information Systems*, 2001, eds. W. Guarino, and Smith, 47–58.
- Knowledge Interchange Format (KIF) Specification (draft proposed American National Standard [dpANS] NCITS.T2/9 8-004: <http://logic.stanford.edu/kif/dpans.html> Now superseded by Common Logic.
- Knowledge Web (KWEB). 2004–2008. <http://knowledgeweb.semanticweb.org/index.html>
- Kokinov, B., D.C. Richardson, T.R. Roth-Berghofer, and L. Vieu, eds. 2007. *Modeling and Using Context 6th International and Interdisciplinary Conference, CONTEXT 2007*, Roskilde, Denmark, 20–24Aug 2007, Proceedings. Lecture Notes in Artificial Intelligence Vol. 4635, 2007.
- Kutz, O., and T. Mossakowski. 2007. *Modules in transition - Conservativity, composition, and colimits*. Second International Workshop on Modular Ontologies. 28 Oct 2007, Whistler, BC.
- Lenat, D. 1998. The dimensions of Context-space, cycorp technical report, 28 Oct 1998. <http://www.cyc.com>
- Lenat, D., and R. Guha. 1990. *Building large knowledge based systems*. Reading, MA: Addison Wesley.
- Lewis, D. 1980. Index, Context, and Content. In *Philosophy and grammar*, eds. S. Kanger, and S. Ohman, Dordrecht: Reidel Publishing.
- Loux, M.J. 2002. *Metaphysics: A contemporary introduction*, 2nd edn. London and New York: Routledge.
- Lüttich, K., C. Masolo, and S. Borgo. 2006. Development of modular ontologies in CASL. Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06, CEUR Workshop Proceedings, vol. 232, eds. H. Haas, S. Kutz, and A. Taminin, 5 Nov 2006, Athens, GA.
- Loebe, F. 2006. Requirements for Logical Modules. Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06, eds. H. Haas, S. Kutz, A. Taminin, 5 Nov 2006, Athens, Georgia, USA. CEUR Workshop Proceedings, Vol. 232.
- Makarios, S. 2006a. A model theory for a quantified generalized logic of contexts. Stanford University Technical Report KSL-06-08. ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-06-08.pdf
- Makarios, S. 2006b. ICL Specification. On behalf of IKRIS Contexts Working Group. nrrc.mitre.org/NRRC/Docs_Data/ikris/ICL_spec.pdf.
- Makarios, S. 2006c. ICL Guide. On behalf of IKRIS Contexts Working Group. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/IKL_guide.pdf
- Mac Lane, S. 1971. *Categories for the working mathematician*. New York: Springer.
- Makowsky, J.A. 1992. Model theory and computer science: An appetizer. In: *T.S.E. Handbook of logic in computer science, Volume 1, background: Mathematical structures*, eds. S. Abramsky, D. Gabbay, and Maibaum, 763–814. Oxford: Clarendon Press.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb Deliverable D18 Ontology Library (final), 31 Dec 2003.
- McCarthy, J., and S. Buvač. 1997. Formalizing Context (Expanded Notes). In *Computing natural language*, eds. A. Aliseda, R. van Glabbeek, and D. Westerståhl, Amsterdam: Stanford University. <http://www-formal.stanford.edu>.
- McCarthy, J. 1987. Generality in artificial intelligence. *Communications of the ACM* 30(12): 1030–1035.
- McCarthy, J. 1990. *Formalizing common sense: Papers By John Mccarthy*. Westport, CT: Ablex Publishing Corporation, 355 Chestnut Street, Norwood, NJ 07648.
- McCarthy, J. 1993. Notes on formalizing context. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.
- Menzel, C. 1999. The objective conception of context and its logic. *Minds and Machines* 9(1):29–56 (Feb 1999).

- Obrst, L., D. Nichols. 2005. *Context and ontologies: Contextual indexing of ontological expressions*. AAAI 2005 Workshop on Context and Ontologies, poster, AAAI 2005, 9–13 July, Pittsburgh, PA.
- Obrst, L., H. Liu, and R. Wray. 2003. Ontologies for corporate web applications. *Artificial Intelligence Magazine*, special issue on Ontologies, American Association for Artificial Intelligence, ed. C. Welty, 49–62, Fall 2003.
- Obrst, L., D. Nichols. 2005. *Context and ontologies: Contextual indexing of ontological expressions*. AAAI 2005 Workshop on Context and Ontologies, poster, AAAI 2005, July 9–13, Pittsburgh, PA. http://www.mitre.org/work/tech_papers/tech_papers_05/05_0903/index.html
- Obrst, L., P. Cassidy, S. Ray, B. Smith, D. Soergel, M. West, and P. Yim. 2006. The 2006 Upper Ontology Summit Joint Communiqué. *Journal of Applied Formal Ontology* 1:2.
- Obrst, L., T. Hughes, and S. Ray. 2006. *Prospects and possibilities for ontology evaluation: The view from NCOR*. Workshop on Evaluation of Ontologies for the Web (EON2006), Edinburgh, UK, 22 May 2006.
- Obrst, L., W. Ceusters, I. Mani, S. Ray, and B. Smith. 2007. The Evaluation of Ontologies: Toward Improved Semantic Interoperability. Chapter In *Semantic web: Revolutionizing knowledge discovery in the life sciences*, eds. J.O. Christopher, Baker, K-H. Cheung, Heidelberg: Springer.
- Obrst, L. 2002. Ontology spectrum and semantic integration and interoperability. Briefing to DARPA, 2 June 2002.
- Obrst, L., and I. Mani, eds. 2000. Proceedings of the Workshop on Semantic Approximation, Granularity, and Vagueness, 11 Apr 2000, Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), 12–16 Apr, Breckenridge, CO.
- Obrst, L., G. Whittaker, and A. Meng. 1999. Semantic interoperability via context interpretation, Context-99, Trento, Italy, Apr 1999, invited poster session.
- Object Management Group (OMG): Meta Object Facility (MOF) Specification. 2006. Version 2.0. Object Management Group (OMG), Needham, Massachusetts
- OntoWeb. <http://www.ontoweb.org/> Finished on 13 May 2004. Follow-on project was Knowledge Web.
- OpenCyc Website. <http://www.opencyc.org/>
- OWL 1.1. Web Ontology Language Tractable Fragments, W3C Member Submission 19 December 2006. <http://www.w3.org/Submission/owl11-tractable/>
- Niles, I., and A. Pease. 2001a. Origins of the IEEE Standard Upper Ontology, in Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology.
- Niles, I., and A. Pease. 2001b. Towards a standard upper ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), eds. C. Welty, and B. Smith, Ogunquit, Maine, 17–19 Oct 2001.
- Partridge, C. 2002. Note: A couple of meta-ontological choices for ontological architectures. Technical Report 06/02, LADSEB-CNR, Padova, Italy, June 2002. http://www.boroprogram.org/bp_pipex/ladsebreports/ladseb_t_r_06-02.pdf
- Phytilla, C. 2002. An Analysis of the SUMO and Description in Unified Modeling Language, Apr 2002, unpublished. <http://ontology.tekknowledge.com/Phytilla/Phytilla-SUMO.html>
- Pianesi, F., and A. Varzi. 2000. *Events and event talk: An introduction*. Ch. 1 of Higgenbotham et al., 3–48. New York: Oxford University Press.
- Poli, R. 2003. Descriptive, formal and formalized ontologies. In *Husserl's logical investigations reconsidered*, ed. D. Fiset, and D. Kluwer, 193–210. Dordrecht: Kluwer.
- Poli, R. 2010. The categorial stance. **Chapter 2** in Part One: Ontology as Technology in the book: *TAO – Theory and applications of ontology, volume 2: The information-science stance*, eds. M. Healy, A. Kameas, and R. Poli.
- Poli, R., and L. Obrst. 2010. The interplay between ontology as categorial analysis and ontology as technology. **Chapter 9** in Part One: Ontology as Technology in the book: *TAO – Theory and applications of ontology, volume 2: The information-science stance*, eds. M. Healy, A. Kameas, and R. Poli.

- Pulvermacher, M., L. Obrst, S. Semy, and S. Stoutenburg. 2005. *Perspectives on applying semantic web technologies in military domains*, MITRE Technical Report.
- Royce, W. 2002. CMM vs. CMMI: From conventional to modern software management. Rational edge: E-zine for the rational community, Feb 2002. Rational Software Corporation. <http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/feb02/ConventionalToModernFeb02.pdf>
- Samuel, K., L. Obrst, S. Stoutenburg, K. Fox, A. Johnson, K. Laskey, D. Nichols, and J. Peterson. 2006. *Transforming OWL and semantic web rules to prolog: Toward description logic programs*. Poster, ALPSWS: Applications of Logic Programming in the Semantic Web and Semantic Web Services, 16 Aug 2006, Federated Logic Conference 2006, Seattle, WA.
- Semy, S., M. Pulvermacher, and L. Obrst. 2005. Toward the use of an upper ontology for U.S. government and U.S. Military domains: An evaluation. MITRE Technical Report, MTR 04B0000063, Nov 2005. http://www.mitre.org/work/tech_papers/tech_papers_05/04_1175/index.html
- Simperl, E.P.B., C. Tempich, and Y. Sure. 2006. ONTOCOM: A cost estimation model for ontology engineering. Proceedings of the International Semantic Web Conference ISWC 2006. <http://ontocom.ag-nbi.de/>
- Smith, B. 2004. Beyond concepts: Ontology as reality representation. In *Proceedings of the Third International Conference on Formal Ontology and Information Systems*, eds. A. Varzi, and L. Vieu, 73–84. FOIS 2004, Turin, 4–6 Nov 2004, Amsterdam, The Netherlands: IOS Press.
- Smith, B., J. Williams, and S-K. Steffen. 2003. The ontology of the gene ontology. *AMIA Annual Symposium Proceedings 2003*:609–613.
- Smith, B. 1996. Mereotopology: A theory of parts and boundaries. *Data and Knowledge Engineering* 20:287–303.
- Smith, B. 1998. Basic concepts of formal ontology. In *Formal ontology in information systems*, ed. N. Guarino, 19–28. Amsterdam: IOS Press.
- Smith, M.K., C. Welty, and D.L. McGuinness. 2004. OWL web ontology language guide, W3C Recommendation, 10 Feb 2004. <http://www.w3.org/TR/owl-guide/>
- Schneider, L. 2003. How to build a foundational ontology: The object-centered highlevel reference ontology OCHRE. In *Proceedings of the 26th Annual German Conference on AI, KI 2003: Advances in Artificial Intelligence, volume 2821 of Lecture Notes in Computer Science*, eds. A. Günter, R. Kruse, and B. Neumann, 120–134. Springer, 2003.
- Schorlemmer, W.M., and Y. Kalfoglou. 2008. Institutionalising ontology-based semantic integration. *Journal of applied ontology*. 3(3):131–200.
- Sider, T. 2002. *Four-dimensionalism: An ontology of persistence and time*. Oxford: Oxford University Press.
- Sowa, J.F. 2005. Theories, Models, Reasoning, Language, and Truth. <http://www.jfsowa.com/logic/theories.htm>
- Standard Upper Ontology (SUO) Working Group Website. <http://suo.ieee.org/>
- Stalnaker, R. 1998. On the representation of context. *Journal of logic language and information*.
- Stoutenburg, S., L. Obrst, D. Nichols, K. Samuel, and P. Franklin. 2006. Applying semantic rules to achieve dynamic service oriented architectures. RuleML 2006: Rules and Rule Markup Languages for the Semantic Web, co-located with ISWC 2006, Athens, GA, 10–11 Nov 2006. In *Service-Oriented Computing – ICSOC 2006, Lecture Notes in Computer Science Volume 4294, 2006, 581–590*. Heidelberg: Springer.
- Stoutenburg, S., and L. Obrst, D. Nichols, J. Peterson, and A. Johnson. 2005. Toward a standard rule language for semantic integration of the DoD enterprise. W3C Workshop on Rule Languages for Interoperability, 27–28 Apr 2005, Washington, DC.
- Stoutenburg, S., and L. Obrst. 2005. Toward a standard rule language for enterprise application integration. Rules and Rule Markup Languages for the Semantic Web, presentation, 3rd International Semantic Web Conference, 7–11 Nov 2005, Hiroshima, Japan.
- Stoutenburg, S., and L. Obrst. 2005. Orchestration of ontologies and rules for integration of the DoD enterprise. Protégé With Rules Workshop, paper and presentation, 8th International Protégé Conference 2005, 18–21 July Madrid, Spain.

- Strawson, P.F. 1959. *Individuals: An essay in descriptive metaphysics*. London: Methuan University Press.
- Suggested Upper Merged Ontology (SUMO) Website. <http://www.ontologyportal.org/>
- Tarski, A. 1933. The concept of truth in formalized languages. In *Logic, Semantics, Mathematics*, ed. J. Woodger, Oxford: Oxford University Press.
- Tarski, A. 1944. The Semantic Conception of Truth and the Foundations of Semantics. In *Philosophy and phenomenological research, Volume 4*, 1944, 341–375. Reproduced in: *The Philosophy of Language*, A.P. Martinich, ed. 48–71. 1985, Oxford: Oxford University Press.
- Thurman, D.A., A.R. Chappell, and C. Welty. 2006. Interoperable knowledge representation for intelligence support (IKRIS) evaluation working group report, 31 Dec 2006. http://nrrc.mitre.org/NRRC/Docs_Data/ikris/IKRIS_Evaluation_Report_31Dec06.doc
- Upper Cyc: <http://www.cyc.com/cycdoc/vocab/vocab-toc.html>
- Upper Ontology Summit, Ontolog Forum, 2006. <http://ontolog.cim3.net/cgi-bin/wiki.pl?UpperOntologySummit>
- Van Leeuwen, J, ed. 1994. *Handbook of theoretical computer science. Volume B: Formal models and semantics*. Amsterdam: Elsevier, and Cambridge: MIT Press.
- Welty, C. 2006. Interoperable knowledge representation for intelligence support (IKRIS). Briefing to ontolog forum, 10 Oct 2006. <http://colab.cim3.net/file/work/SICoP/2006-10-10/Presentations/CWelty10102006.ppt>
- Welty, C., and D. Ferucci. 1999. Instances and classes in software engineering. *Artificial intelligence*, 24–28. Summer, 1999.
- What's in Cyc? http://www.cyc.com/cyc/technology/whatis_cyc_dir/whatsincyc
- Williamson, Timothy. 1998. *Vagueness*. London, New York: Routledge.
- WonderWeb Website. <http://wonderweb.semanticweb.org/objectives.shtml>. Completed, July 2004

Chapter 3

Organization and Management of Large Categorical Systems

Frank Loebe

3.1 Introduction

Ontologies grow large in size if systems are considered that cover a very complex domain like medicine, or a number of domains possibly together with upper levels of categories. Especially for philosophical investigations of basic categories, the ontological systems studied in the past typically comprise only a handful of basic categories, as in the works of Aristotle, Kant, Brentano, Husserl, and many others (see Sowa, 2000, Section 2.2 for an overview in the context of computer science). A proper analogy holds for logic, considering the limited size of axiomatic systems formerly under examination. Manual logical investigations were usually concerned with studying the consequences of a few axioms, a task hard enough itself. Trying to model real-world problems with logic, even when equipped with automated reasoning as currently available, one easily faces tremendously large signatures and theories which require novel solutions. A few numbers may provide some intuition on the current scale of the problems. As a self-contained system, the leading clinical healthcare terminology system SNOMED CT is implemented in a logical formalism and comprises more than 311,000 concepts (with formal definitions) as of January 2008.¹ On this scale of even a single system, new methods are required for applying formalisms and using available technologies. The situation is even more complex in the context of the Semantic Web effort (Berners-Lee et al., 2001). Hendler and van Harmelen (2007, p. 823) qualifies its major representation languages RDF, RDFS (W3C, 2004a) and OWL (W3C, 2004b) as “the most widely used KR (knowledge representation) languages in history” and states that “[a] web search performed around the beginning of 2007 finds millions of RDF and RDFS documents, and tens of thousands of OWL ontologies.” Many of those OWL ontologies

F. Loebe (✉)

Department of Computer Science, University of Leipzig, Leipzig, Germany
e-mail: frank.loebe@informatik.uni-leipzig.de

¹Since 2007, SNOMED CT, the Systematized Nomenclature of Medicine – Clinical Terms, is maintained and further developed by the International Health Terminology Standards Development Organization (IHTSDO), see <http://www.ihtsdo.org>. The number originates from the page <http://www.ihtsdo.org/snomed-ct/snomed-ct/>, accessed on 13.09.2008.

complement one another regarding the domains which they cover. Consequently, the use of several such ontologies as components of a joint, larger system is an immediate idea. However, the first detailed attempts to elaborate this idea have discovered many unresolved issues and choices for solutions.

In this chapter we survey approaches to tackle the complexity of huge categorical systems and describe the state of the art from two perspectives: (1) the internal organization and use of categorical systems, and (2) modularization issues of logical formalisms. Put more generally, (1) focuses on a single-category perspective, i.e., it is concerned with means for specifying or searching single categories in the system. In contrast, (2) involves a perspective of the system as a whole, as typically held during the construction and maintenance of categorical systems. For (1), we start from precursors of current formal ontologies by reviewing work in terminology and classification sciences, describing the move toward the use of logical approaches and their benefits. The main part is then concerned with aspect (2) and discusses modularization universally as well as applied to formal logical languages. Since modularization as a field is still emerging, this part sets up a terminology of modules and module characteristics in order to describe and evaluate the current branches of research to some extent uniformly.

The present chapter assumes a very broad notion of ontology. It includes all kinds of concept systems, i.e., systems aiming at specifying a collection of concepts or notions, which themselves are referred to in other information systems. Hence, natural language texts serve as presentations of ontologies just as other types of systems, ranging by increasing degree of formality from vocabularies, glossaries, and thesauri over terminologies or conceptual models up to formal axiomatizations, to name just a few types (Gómez-Pérez et al., 2004, ch. 1). In a sense, our scope is similar to that in terminology management (Wright and Budin, 1997), but with less consideration of representational or linguistic aspects like the naming of concepts (cf. also de Keizer et al., 2000).

Terminological and ontological aspects are relevant to all domains, and terminological efforts in the form of determining a common language arise immediately in most fields of science and engineering. In the next section, we concentrate on the development of concept systems in medicine as a representative of domains with high complexity.

3.2 Terminological Systems in Medicine

The development of common vocabularies and classification systems in medicine started very early, exemplified by the introduction of the International Statistical Classification of Diseases (ICD) in 1893 (WHO, 2004). Concept systems in medicine grow very large as soon as a broad coverage of subdomains is to be combined with even a medium depth of subsumption hierarchies.² Medical

²Most of the current terminological systems arrange concepts by subsumption relations, i.e., along their degree of generality. The broad use of the term “subsumption hierarchy” includes trees and

terminology management has thus developed from mere collections of terms (and in some cases definitions) in the form of vocabularies, glossaries, classifications, code lists, nomenclatures, and others via semi-formal models to logic-based approaches (cf. Herre, 2010b) in this volume or de Keizer et al. (2000). We briefly review major aspects in the course of that development in order to identify solved and unsolved problems concerning the complexity of these systems. Nevertheless, this will ignore most of the overwhelming number of problems relating to clinical terminology (Rector, 1999; Cimino, 1998), many of which still remain open.

The move toward formal systems is primarily driven by two partially interrelated aspects, which are often intermingled in the literature: (1) *compositionality* and (2) the *structured arrangement of concepts*.³ Both are motivated by mitigating the combinatorially exploding number of concepts in medicine. For instance, given the notions of fracture as well as of bone and bone subtypes, there are fractures of many kinds, like fractures of the arm, the forearm, the ulna, the radius, the femur, etc., which need to be available e.g. for electronic health records.

3.2.1 Compositionality

Looking at aspect (1), semi-formal compositional concept systems try to reduce the representational complexity by providing a number of atomic concepts, e.g., fracture and arm, forearm, ulna, and radius. *Compositionality* means to express certain concepts in terms of combinations of others, starting from atomic concepts. Expressions of the form “(fracture, ulna)” are used for representing the notion of “fracture of the ulna”. The number of explicitly managed concepts in a compositional setting can be greatly reduced to the number of atomic concepts. Compositionality is in this context also called *post-coordination* and contrasted with *pre-coordination* (cf. Rector, 1999, p. 246). Pre-coordinated terminological systems are constructed as pre-established enumerations of concepts, listing all available concepts explicitly, whereas post-coordinated systems allow for forming implicit concepts from explicitly given constituents. Both approaches usually aim at a degree of coverage as high as possible, and at internal consistency of the resulting concept system.

Two new problems arise for compositional systems: (a) meaningless combinations of concepts and (b) the need to detect equivalent combinations. An example for (a) is (fracture, oral cavity), which is not reasonable because cavities cannot break. The pairs (fracture, ulna) and (ulna, fracture) illustrate problem (b), if they are understood to express the same concept. Both problems were tackled by the use of description logics (Baader et al., 2003b) starting in the 1990s, for instance, in GALEN (Rogers, 2001; Rector and Rogers, 2005) and SNOMED CT (Iggulden and Price, 2001). In particular, equivalence and subsumption checking of

directed acyclic graphs, also called polyhierarchies. The depth of such hierarchies refers to the path lengths between roots and leaves.

³This mixture appears, e.g., in the description of first-, second- and third-generation terminologies; (cf. Rossi Mori, 1997; Spackman and Campbell, 1998; Straub, 2002).

concepts are common reasoning problems in description logic (Baader et al., 2003b, Section 2.2.4) which address (b). Problem (a) of meaningless concept definitions can only partially be supported by a description logic formalism. For this purpose, concepts and relations among them must be expressed appropriately and need to be augmented by restrictions. For example, (fracture, ulna) could be represented adequately as $\text{fracture} \sqcap \exists \text{has-location. ulna}$. Description logics can further be used to detect inconsistencies among logical restrictions, i.e., they support ensuring consistency. Moreover, less common reasoning tasks like determining the least common subsumer of a given set of concepts or the most specific concept of an instance can help in defining new concepts (cf. Baader et al., 2003b, Sections 1.6 and 6.3). Recently, rather weakly expressive description logics – called the EL family – are gaining much attention, including their theoretical foundations (Baader, 2003a; Brandt, 2004; Baader et al., 2005; Krisnathi and Lutz, 2007) as well as efficient reasoners (Baader et al., 2006). One major reason for this development is the (re)discovery that the limited expressivity is already beneficial for large medical terminologies like GALEN and SNOMED CT, while more expressive logics do not yet scale to systems of this size.⁴

Although meanwhile superseded by description logics, it is informative to look at earlier intermediate solutions to the problem of arbitrary combinations, namely the use of *multi-dimensional* or multi-axial concept hierarchies introduced in the late seventies (cf. Spackman and Campbell, 1998). The restriction to combine only concepts from different dimensions like anatomic site, morphology, or etiology avoids insensible combinations along one and the same axis, e.g. (ulna, femur). Since such restrictions remain insufficient, the approach has been refined to *multi-focal* models (cf. Straub, 2002). Here, not all dimensions are treated equally, but certain dimensions appear – possibly constrained in their values – attached to specific values at another dimension, where a value spanning its own field of dimensions is called a *focus*. For instance, the value “fracture” on a “disease” dimension may be a focus with a “location” dimension constrained to “bones”.⁵

⁴GALEN and SNOMED CT have adopted weak description logics very early. Spackman and Campbell (1998) and Spackman (2001) report the use of a very restricted and therefore computationally well tractable description logic for an earlier version of SNOMED CT. The only concept constructors referred to are conjunction and existential restrictions, plus top and bottom symbols (apart from bottom, these constructors form nowadays the EL description logic (Baader, 2003a)). Such usage is also claimed for GRAIL, the language used for GALEN. However, according to Rector et al. (1997), the structure of GRAIL appears to be related to, but somewhat different from standard description logics; (cf. also Rector and Rogers, 2005, p. 12 f.).

⁵Straub (2002) further proposes so-called *multi-point models* in order to allow for multiple values on one and the same dimension on a regulated basis, motivated by examples like a double fracture and given an analysis why extensions by another dimension would not solve that representation problem. Though we agree on the examples, some skepticism about this solution remains on our side, but cannot be elaborated here.

3.2.2 Navigation

Although a number of systems based on multi-dimensional or multi-focal models are currently found in practice (e.g. LOINC (Forrey et al., 1996; McDonald, 2003)), description logics prevail with regard to their combinatorial capabilities and the validation of the consistency of restrictions.

Nevertheless, those former approaches have an interesting effect concerning the organization of subsumption hierarchies, aspect (2) from above. Subsumption reasoning in description logics allows for inferring formally consistent subsumption lattices based on concept definitions. However, the number of explicitly introduced elements in these lattices usually becomes very large, due to naming composed concepts in order to reuse them without recomposing them in each case of use. In spite of their inferential potential, current logical approaches and tools do not yet offer support to tackle the *navigation problem*, i.e., the problem of comprehending and orienting creators and users of such lattices.⁶ The availability of corresponding methodological advices is equally limited. Alan Rector is among the few authors addressing this question in the field of computer science (Rector, 2003). His approach to organizing concepts is similar to the main idea of multi-dimensional systems: for the construction of large concept systems one starts with a number of initially unrelated, self-contained, even mono-hierarchical taxonomies, called *primitive skeletons*. Those hierarchies should further be homogeneous, i.e., the classification within a primitive skeleton follows a single criterion (or several progressively more restrictive criteria). Once primitive skeletons are established, all remaining concepts are defined by means of logical definitions using skeleton concepts, which yields the effect that any concept with multiple parents is a defined concept.

Rector's approach is not only reasonable for the construction phase of an ontology, but having access to these distinctions in the overall system can further be exploited for navigational purposes. Over the last decade, methods corresponding to the multi-dimensional organization of concepts are studied in information retrieval as *faceted browsing/search* or *dynamic taxonomies* (Sacco, 2000, 2006). A major focus in this area is the usability of systems. For instance, Hearst (2006) suggests design recommendations for faceted search interfaces and reports that the use of multiple hierarchies does not confuse users. Faceted browsing is further gaining popularity in the context of the Semantic Web (Hildebrand et al., 2006; Specia and Motta, 2007), and it relates closely to *faceted classification* in library and information sciences. One of the initial proponents of the latter is Shiyali R. Ranganathan, who elaborately provides methodological guidance on the construction of faceted systems in this field (Ranganathan, 1967, 1962; cf. Spiteri (1998) for simplifications). From a very general point of view, characteristics of facets may be found in many representation approaches, as argued in Priss (2000), which makes a first,

⁶See also Dzbor and Motta (2008) for the navigation problem and related challenges from the perspective of human-computer interaction applied to ontological engineering.

integrative attempt to abstractly describe and formalize facets. However, an in-depth elaboration of facets, neither formally nor ontologically, does not seem to be available yet, albeit it appears desirable due to its ascribed potential of enhancing formal approaches, including description logics.

Intermediately summing up, the complexity of concept systems in terms of the number of their concepts has primarily been addressed by description logics. This solves the larger parts of the compositionality issues, including support for ensuring consistency and to some extent for avoiding insensible concepts. But with respect to search and navigation within large systems further improvements are desirable and may be transferred from other fields.

3.3 Complex Systems and Modularization in General

The previous section is primarily concerned with a single-concept perspective on the overall system, which is a common case from a user point of view. There is another aspect of modularity which is concerned with the overall structure and architecture of an ontology. This trespasses the view of ontologies as a huge set of interrelated concepts or, more formally, as a very large set of formal sentences, which becomes highly relevant for the construction, maintenance and evolution of large ontologies. In addition, modularity contributes to the comprehension by maintainers and users, possibly to training the latter in using the system, and it might correspond to some extent to mental knowledge organization principles. Many systems in nature are very large and complex, which requires major theoretical efforts in order to comprehend and describe them. Artificially created systems have reached similar degrees of complexity. *Modular design* is a universal approach to alleviate complexity in terms of artefactual systems, which is employed in every engineering discipline, e.g. from construction to machine to software engineering. Independent of the reduction of complexity and better comprehensibility, further functional advantages of modular design are the facilitation of change and the encouragement of the parallel development of different parts of a system.

Focusing on ontologies, *modularity* is a very young research area in their respect. Most (medical) terminologies are sparsely structured in their models (Gangemi, 1999, p. 190).⁷ The recency of modularity for ontologies applies particularly to ontologies as they appear in currently popular application fields. In the Semantic Web (Berners-Lee et al., 2001), for instance, there is a growing need and discussion on modularization of web ontologies, exemplified by a new series of workshops on modular ontologies (Haase et al., 2006; Cuenca Grau et al., 2008a; Sattler and Tamilin, 2008). Much work there is devoted to providing modularity regarding the Web Ontology Language (OWL) (W3C, 2004b). Bio-ontologies such as the Gene

⁷On the technical level, they are most often delivered in the form of huge data files, with a technically oriented structure which is rather independent of the conceptual architecture.

Ontology (Ashburner et al., 2000) or the Open Biomedical Ontologies (OBO)⁸ form another active area of application (Kelso et al., 2010, this volume).

There are a number of domains providing ideas and initial approaches for modularization of ontologies. An active field highly intertwined with it is ontology integration or ontology matching (cf. also Kalfoglou and Schorlemmer, 2010) in this volume. Ontology integration is to some extent more mature than modularization, with Kalfoglou and Schorlemmer (2003) and Noy (2004) presenting first review articles of the subject, and a comprehensive book (Euzenat and Shvaiko, 2007) being recently available. Moreover, a couple of approaches in knowledge representation offer accounts which may be reused for modularization. The area of knowledge-based systems has also developed solutions – typically added on top of the basic representation formalisms employed. Looking at implemented approaches, an often-mentioned parallel for managing large formal systems, including concept systems, is software engineering (Loebe, 2006; Amir and McIlraith, 2005; Diaconescu et al., 1993). In connection with formalisms, modularization is frequently studied with respect to formal structures rather than the contents expressed. We follow this route in reviewing and discussing logical theories and modularization in the remainder of this chapter.⁹

Apart from high-level functional desiderata for logical modules, such as facilitating reuse or maintainability, a clear notion of module has not yet been established. In the context of OWL, the primary ontology language of the Semantic Web which is based on rather expressive description logics (Baader et al., 2003b), only a simplistic means is available to tie formulas together which are syntactically distributed over several files. The owl:imports statement is the only element available for a syntax-level splitting of OWL ontologies. That corresponds to the union of the axioms of the importing ontology and the imported ones, which is insufficient from the functional perspective outlined above. Meanwhile, first approaches have been developed in this context, with differing aims and outcomes (e.g. Bouquet et al., 2003; Cuenca Grau et al., 2006c, 2007a; Kutz and Mossakowski, 2008). In most cases they are based on earlier work which proves adaptable to modularity issues, and usually they fall into one of two types, distinguished by the question of how a resulting modular system is constructed. On the one hand, a *compositional* route can be taken by defining modules and general ways of how these combine into systems. On the other hand, several *decompositional* approaches consider the problem of finding appropriate ways of partitioning a given, typically large ontology. In order to interrelate approaches of either type via a uniform terminology, we formulate a self-contained, general framework in the next section.

⁸Open Biomedical Ontologies: <http://obo.sourceforge.net/>

⁹Terminologically, we separate the use of “ontology” from that of logical “theory”, generally adhering to more formal, logical vocabulary in this part of the chapter. In the same line the term “semantically” should now be read as referring to a formal, model-theoretic semantics.

3.4 Abstract Framework for Modules

3.4.1 Overview

The purpose of this framework is to achieve a degree of comparability among the families of current and relevant earlier approaches that are reviewed in Section 3.6. Modules are only introduced on a very general level in this section. Section 3.5 presents a cross-cutting view on the literature by collecting a number of specific characteristics that are applied in Section 3.6.¹⁰

Basically, modules are understood as components of systems which contain information and are interconnected at interfaces (as known from software engineering), where they can exchange information. The primary elements of this model are illustrated in Fig. 3.1. More logically speaking, such system components contain logical theories, i.e., sets of sentences, which they exchange through their interfaces. Moreover, the system itself can influence this exchange among modules by means of a composition operation.

In the subsequent sections, we refine these intuitions and capture them more precisely in conventional mathematical style, for general comprehensibility.¹¹ As an example domain, the modular construction of a top-level ontology is utilized, primarily focusing on the combination of a module for time with a module for processes.

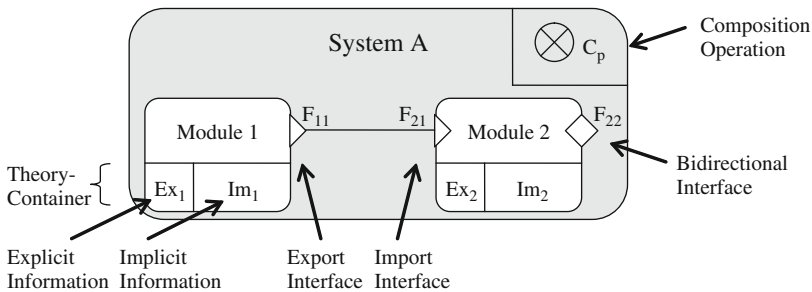


Fig. 3.1 Illustration of major conceptual components of the abstract module definition

¹⁰Note that Section 3.6 provides a review of the selected approaches which should to a large extent be readable without detailed knowledge of the framework. However, the latter supports a unified view and collects recurrent properties required for modules in Section 3.5.

¹¹Note that partially similar issues are treated in Kutz and Mossakowski (2008) at a comparable level of generality, exposed in more sophisticated terms on the basis of category theory; cf. Section 3.7.1 for more details about this and related works.

3.4.2 Formal Preliminaries

In the sequel, a very general understanding of a logic is used in order to cover a range of approaches, along the lines of Barwise and Feferman (1985) and Herre (1995). A logic \underline{L} is understood as a triple $\underline{L} = (L, MS, \models)$ of a language L , considered a set of sentences,¹² a set of model structures MS , and a satisfiability relation $\models \subseteq MS \times L$ between model structures and sentences. The latter generalizes in the standard way¹³ to model and consequence relations between models, model classes, and theories, also denoted by \models . The powerset of an arbitrary set S is denoted by $Pow(S)$.

Grammar-based definitions of a language L usually distinguish between sentence constructions and a signature/vocabulary V . $Lg(V)$ designates the language over V , i.e., all sentences which can be constructed from V in a given grammatical framework. Sometimes it is useful to speak of the same language even for different vocabularies. Any subset $T \subseteq Lg(V)$ is called a theory. $Voc(T)$ denotes the vocabulary on which T is based, the language of T is $Lg(T) =_{df} Lg(Voc(T))$. For $T \subseteq Lg(V)$ and $V' \subseteq V$, the restriction of T to V' is defined as $T|_{V'} =_{df} T \cap Lg(V')$. $Cn(T)$ denotes the deductive closure of a theory T regarding a fixed consequence relation \models . $Cc(T)$ refers to the set of classical consequences of T , in contrast to certain non-monotonic consequence relations, for instance. The use of Cc assumes that T is formulated in a language which can be interpreted classically. The set of all tautologies of a logic is referred to as $Taut(V)$ for a signature V and $Taut(L)$ for a language L . This formal setting is broad enough to cover at least classical propositional logic (PL) and first-order logic (FOL), as well as description logic (DL) and a number of rule-based approaches with a model-theoretic semantics. Moreover, many non-monotonic formalisms likewise fit under this umbrella.

3.4.3 Defining Modules

Common to all modules under consideration is that they contain information, i.e., formally, they “contain” a theory. Everything from which a theory may be extracted is called a *theory container*. $Th(C)$ denotes the theory within a theory container C . Further, two kinds of sentences in a theory are distinguished, motivated by the difference between axiom sets and deductively closed theories: sentences may belong *explicitly* or *implicitly* to a theory, where implicit sentences are in some sense derived. For instance, an axiomatization of time may form the explicit part of a theory container, whereas consequences of that axiomatization which are not axioms themselves pertain to the overall theory implicitly.

¹²We consider only closed formulas, i.e., formulas without free variables (in languages with variables).

¹³One may consider first-order logic as a prototypical case for those notions; (cf. Ebbinghaus et al., 1994, ch. III).

The set of explicit sentences within a theory container C is denoted by $Ex(C)$, implicit sentences by $Im(C)$, which do not overlap and together yield $Th(C)$, i.e., $Ex(C) \cap Im(C) = \emptyset$ and $Th(C) = Ex(C) \cup Im(C)$. Note that a theory container C may distinguish explicit and implicit parts of its theory without the implication that the latter is deductively closed; hence $Th(C) \neq Cn(Th(C))$ remains possible.

An essential aspect of modules is that they are combined and used together, resulting in a larger system and module intercommunication. Adapted from software engineering, the notion of interfaces is very relevant in this respect. Loebe (2006) discusses a number of options for interpreting the notion of interface for the logical case, among them the view of interfaces as theories (which should be extracted from the module theory). Here we refine this view and distinguish interfaces and their specifications, where the former arise from applying the latter to a theory container. To illustrate the intentions for these notions, consider a time theory which may comprise an elaborate axiomatization of time points, intervals and various relations among them. An interface may be intended to provide restricted access to that theory in terms of a language restriction, e.g. to a theory of intervals and the part-of relationship between intervals only. The interface specification defines that restriction abstractly, and an interface is created by applying the restriction to the time theory, i.e., to a particular theory container.

More formally, an *interface specification* $FS = (L_M, L_F, Op_F)$ is defined by two interfaced languages, L_M internal to the module and L_F as available externally at the interface, and a transformation operation Op_F . This operation may serve purposes of adapting internal formulas of the module for external combinations, e.g. by additional relativization of quantifiers (cf. Ebbinghaus et al., 1994, ch. VIII, p. 119 ff.). It may likewise not change anything in the special case that Op_F is the identity operation.

Interfaces are connected with information flow and its direction, which can be realized by an exchange of formulas among two theory containers via a connection among their interfaces. We distinguish three types of interface specifications: *import* and *export* interface specifications as well as *bidirectional* ones. Import (export) interface specifications are connoted with the expectation that formulas in the interface language (a) are primarily provided outside (inside) the interfaced theory container and (b) they flow only to (away from) the module. In the top-level ontology example, the time theory may provide an export interface for communicating formulas to the process theory at an appropriate import interface. In contrast to strict export and import, bidirectional interfaces allow for the exchange of formulas in both directions, such that assumptions on the mutual competence are less adequate. This requires different operations for these types: import specifications supply $Op_F: Pow(L_F) \rightarrow Pow(L_M)$ and export specifications $Op_F: Pow(L_M) \rightarrow Pow(L_F)$. In the bidirectional case, $Op_F: Pow(L) \rightarrow Pow(L)$ with $L =_{df} L_M \cup L_F$ and $Op_F(T) = Op_F(Op_F(T))$ for every $T \subseteq L$.¹⁴

¹⁴This requirement is a real restriction compared to replacing a bidirectional interface with an arbitrary pair of an import and an export interface.

If an interface specification $FS = (L_M, L_F, Op_F)$ is applied to a theory container C this yields an *interface* F , which is represented as $F = (FS, C)$, where $Cont(F) = C$ and $type(F) \in \{im, ex, bi\}$ as derived from the type of Op_F in FS . The language internal to the module is denoted by $Lgm(F)$, hence $Lgm(F) = L_M$. $Oprn(F)$ designates the operation of the interface specification of F . If an input G is provided to Op_F this yields the theory of the interface, $Th(F) =_{df} Op_F(G)$. For export interfaces, the input is the theory of the container, hence $Th(F) = Op_F(Th(C))$. Resuming the time example, an export interface F defined on intervals and their part-of relationships could result in a subtheory of the elaborated time theory $Th(C)$, possibly with additionally relativized quantification in $Th(F)$.

Transforming a theory container C into a module M means to add a number of interfaces to C , and to provide an operation describing how information imported via interfaces together with explicit/local information of C yield the implicit theory of the module. E.g., in the case of the process module, this operation defines how the process axiomatization combines with imported formulas in the language of time and possibly others. Hence, a module $M = (L_M, C, (F_j)_{j \in J}, Op_M)$ must define $Op_M: Pow(L_M) \times \prod_{j \in J} (Pow(Lgm(F_j))) \rightarrow Pow(L_M)$, which further is the basis to explain the consequences of changes in modules from which a given module imports formulas. The theory of the local container C of M is considered the explicit part of M , if M is viewed as a theory container itself, denoted by $Ex(M) =_{df} Th(C)$, thus requiring $Lg(C) \subseteq L_M$. $Ifc(M) =_{df} (F_j)_{j \in J}$ denotes all module interfaces. Due to Op_M , $Th(M)$ is dependent on the inputs to import and bidirectional interfaces of M , hence using $Th(M)$ must assume a fixed environment/system in which the module is employed, as introduced next. The theory resulting from importing only empty theories is designated as $Th^o(M)$.

The final step is to compose systems from modules, like forming a top-level ontology from the time and the process theory and others. First of all, such a system S will also contain a theory T in a language L . T is usually derived from the modules $(M_k)_{k \in K}$ and the way they are interconnected at their interfaces; below $Ifm((M_k)_{k \in K}) =_{df} \cup_{k \in K} Ifc(M_k)$ refers to all interfaces in a family of modules, which for a system S over that family of modules is abbreviated as $Ifm(S)$. We identify two aspects involved in linking modules and the composed system: (1) a structure U of the intended (possibly mutual) use among the modules, meeting at their interfaces, and (2) a composition operation Cp capturing system-driven transformations between two connected interfaces based on the module usage structure. Cp allows the system to influence the communication between interfaces independently from its modules. For instance, if multiple modules of a top-level ontology provide time formulas, Cp may be used for conflict resolution, taking into account the overall system. Much more commonly, however, systems are formed from axiomatized modules by their set-theoretical union.

A *modular theory* or *system* S is described as $S = (L, (M_k)_{k \in K}, U, Cp)$, where $U \subseteq Ifm(S) \times Ifm(S)$ such that interfaces are connected with suitable antagonists, i.e., for every $(x, y) \in U$: either $(type(x) = bi \text{ or } type(y) = bi)$ or $(type(x) = im [ex] \text{ iff } (type(y) = ex [im]))$. The condition that for every $k \in K: Lg(Th(M_k)) \subseteq L$ is not required for generality, but we expect it to be relevant in many scenarios. Cp

should be a generic operation (i.e., it does not depend on specific modules) and it should respect the distinction of import and export interfaces and the structure U . The theory of S is defined as $Th(S) =_{df} Cp((M_k)_{k \in K}, U)$. A module interface in S is said to be *open* iff it is not connected with any other interface in U . A system is called *saturated* if it does not require external input, i.e., iff there are no open import interfaces; bidirectional interfaces may remain open. Like modules, modular theories are theory containers and may divide their theories into explicit and implicit parts, and they can also be equipped with their own interfaces, independent of those of their modules.

3.4.4 Example Module Types

3.4.4.1 Basic Modules

Two formalisms may provide further illustration of how this framework can be used to understand approaches involving logical modules. The first defines a very common view based on classical logical theories (e.g., in FOL or any DL), considering the union of these theories plus its deductive closure as the composition operation for the system.¹⁵ In the above framework, this corresponds to forming a trivial module out of a theory $T \subseteq L$ by setting $M = (L, T, F, \cup)$ with $F = ((L, L', id), M)$, where id denotes the identity operation, and $L' \subseteq L$ forms the interface language. That means, the theory itself (or parts of it) serves as a bidirectional interface, and obviously there may be several such interfaces for different sublanguages. Modules of this type are called *basic modules*. They also cover cases where parts of the signature are marked as “external” while the information flow remains unrestricted when joining theories (Cuenca Grau, 2007a). Possibly this corresponds to interface languages which are a proper subset of L . Given a number of basic modules $(M_i)_{i \in I}$ and their corresponding family of interfaces $Ifm((M_i)_{i \in I})$, the use of the set-theoretical union as composition produces the system $S = (Lg(\cup_{i \in I} Voc(M_i)), (M_i)_{i \in I}, Ifm((M_i)_{i \in I}) \times Ifm((M_i)_{i \in I}), \cup)$.¹⁶ Consequently, $\cup_{i \in I}(Th(M_i)) \subseteq Th(S) = Cc(\cup_{i \in I}(Ex(M_i)))$. For a logic with Craig interpolation such as FOL (Chang and Keisler, 1990; Craig, 1957), $Th(S)$ is consistent if and only if each pair of connected interfaces is consistent (cf. also Amir and McIlraith, 2005).

¹⁵In the sequel, following common conventions and despite actually distinguishing theories and deductively closed theories, we abbreviate this composition operation as “union” or “set-theoretical union”, denoting it as \cup .

¹⁶Concerning the usage structure among basic modules, there is some arbitrariness between two options: either to consider all interfaces connected to each other, or to see connections only if there are non-empty intersection languages among the M_i . We chose $Ifm((M_i)_{i \in I}) \times Ifm((M_i)_{i \in I})$ because even with empty intersection languages some interaction may occur among the M_i , e.g., creating inconsistency based on logical sentences restricting the universe to different cardinalities. However, for FOL the second option is in effect equivalent to this Amir and McIlraith (2005).

3.4.4.2 Modules in Distributed First Order Logic

Aiming at an uncommon composition operation, another example is provided by Distributed First Order Logic (DFOL; Ghidini and Serafini, 2000), see also Section 3.6.5. DFOL theories are defined in disjoint first order languages, e.g. $L_1 \cap L_2 = \emptyset$. Syntactically, they can be linked with “interpretation constraints” $\varphi \rightarrow \psi$, where φ and ψ belong to different languages, e.g. $\varphi \in L_1$, $\psi \in L_2$. The intuition is that φ in the L_1 -module yields that ψ holds in the L_2 -module.¹⁷ Accordingly, a DFOL interpretation constraint implicitly defines two contributions: one to an export interface F_{ex} such that $\varphi \in Th(F_{\text{ex}})$ and the other to an import interface F_{im} with $\psi \in Th(F_{\text{im}})$, where it is reasonable to let the “union” of all equally directed interpretation constraints between the same modules jointly define those interfaces. Further, the above constraint influences the composition operation of the combined system S by adding the pair $(F_{\text{ex}}, F_{\text{im}})$ to its usage structure, thus contributing to deriving the theory $Th(S)$. Notably, the language of S is restricted compared to a system composed of basic modules. DFOL does not allow for composing formulas with constituents from the different module languages.

Most of the approaches to be studied in Section 3.6 deal with basic modules. However, in general, composition within a system may depart from set-theoretical union. For instance, a system might even provide a weaker theory than each or some of its modules, i.e., $Th(S) \subset Th(M_i)$ for every or some $i \in I$. A selection of general characteristics is presented in the next section, as a foundation for comparison.

3.5 Characteristics of Module Notions

All of the subsequent characteristics have been collected within the literature or have been devised from work on the top-level ontology General Formal Ontology (Herre et al., 2006; Herre, 2010a, this volume). Accordingly, they represent a collection of features or desiderata to which particular notions of module may adhere, instead of a set of necessary requirements for every module definition.

3.5.1 Informal Characteristics

First, we formulate a number of requirements which are hard to state on a sole formal basis.

CI-1 *Comprehensibility*: In order to support maintainability, a module should remain “comprehensible”. Two options in order to achieve this for basic modules are (a) the restriction to a rather small vocabulary and a small set of

¹⁷Interpretation constraints are a special kind of “compatibility constraints” as mentioned in Section 3.6.5. Their proof-theoretic counterparts share the same intuition and are called “bridge rules”, a new type of inference rule in DFOL.

axioms of arbitrary form, or (b) to have a possibly large, structured vocabulary equipped with only simple axioms, which furthermore follow one or a few common schemes. Regarding systems, comprehensibility should derive from the way in which they are composed of basic modules.

CI-2 *Stability*: For a system, changes to a single module or the addition of loosely related modules should not exhibit a strong influence on the structure of the system, i.e., the previous system structure should remain stable. For example, adding a module should not alter connections among other modules. Furthermore, side effects of evolution should be reduced to a minimum. This criterion has also been proposed in Rector (2003).

CI-3 *Compositionality*: For a number of logical properties of module theories it is desirable to be able to reason over the properties of the theory of the resulting system. For example, a composition operation could be devised such that the consistency of a system immediately results from the consistency of its modules. Compositionality would be very valuable especially for intractable logics such as FOL, because one could then concentrate on proving properties of much smaller theories. For more tractable logics, compositionality would still be helpful in practice, because large ontologies can take current reasoners far beyond their capabilities (cf. Pan, 2006).

CI-4 *Directionality* Composition should allow for a directed information flow among modules, such that a module B may use another, A, without B having an impact on A (cf. Ghidini and Serafini, 2000; Bao et al., 2006a). Formally this is possible in the framework by using import and export interfaces. Let F_i and F_k be interfaces with which two components M_i and M_k within a system S are connected via an export-import-connection. The exporting module cannot receive formulas over that connection. However, in order to enforce directionality more generally and also on the system level, composition must additionally assure that the combination of $Th(M_i)$ and $Th(M_k)$ to $Th(S)$ respects this limited information flow.

3.5.2 Formal Characteristics

The remaining features can be formally captured in the framework introduced above. Note that any pre-established constraints with respect to feasibility or computability are not imposed. We will frequently refer to a system S composed of a family of modules $(M_i)_{i \in I}$, for an arbitrary index set I .

3.5.2.1 Characteristics Primarily Based on Either Interfaces, Modules or Systems

CF-1 *Basic Interface and Identity Interface*: Naturally, interfaces should only provide access to and from the module contents. Often, one would even require that they do not change im- or exported formulas. Hence, an interface F is called *basic* iff, for arbitrary input G and output language L , its operation satisfies that $Th(F) \subseteq G|_L$. It is called an *identity interface* iff $Th(F) = G|_L$.

CF-2 Black-box Interfaces: The next characteristics aim at the potential of a module at an interface to interconnect with other modules. It is derived from Cuenca Grau (2007a) and refers to the assumptions a module holds at an interface it imports from. Given a module M with $Ifc(M) = (F_n)_{n \in \mathbb{N}}$, an import or bidirectional interface $F_m = (FS, M)$, $m \in \mathbb{N}$, has a (*simple*) *black-box property* iff for an arbitrary input to F_m such that its theory $Th(F_m)$ is consistent, $Th(F_m)$ can be consistently combined with the theory of otherwise only empty imports to M . In general, that means that for an arbitrary theory T and the family $(T_n)_{n \in \mathbb{N}}$ with $T_n = Oprn(F_n)(\emptyset)$ for $n \neq m$ and $T_m = Oprn(F_m)(T)$ such that T_m is consistent, it follows that $Th(M) = Op_M(Ex(M), (T_n)_{n \in \mathbb{N}})$ is consistent. For basic modules, this simplifies to enforcing that $Th(F_m) \cup Th^\emptyset(M)$ is consistent for arbitrary consistent inputs to F_m . A much stricter version is defined by the analogous assumption for arbitrary consistent imports at all other interfaces of M , which we call a *strict black-box interface* of M .

CF-3 Module Language Overlap: This feature refers to the question of whether module signatures or languages may overlap, thus analyzing $Voc(M_i) \cap Voc(M_j)$ and $Lg(M_i) \cap Lg(M_j)$. There is a variety of options, ranging from empty intersections in both cases¹⁸ over sharing restricted vocabulary elements (e.g. special relations only) to arbitrary overlap among modules.

CF-4 Classically Closed Composition: Due to the important role of classical logics, it is often convenient if systems are closed under classical logic: $Th(S) = Cc(Th(S))$. Basic modules and their compositions are classically closed by definition. The rationale behind this closure is to assure correct reuse of information from S within classical reasoning systems, i.e., when using S itself as a module in another system.

3.5.2.2 Characteristics with Respect to the Interplay of Modules and Systems

For the following characteristics, we assume that modules are equipped with identity interfaces only.

CF-5 Inclusion of Explicit Module Information: It appears natural that a system S should provide the information contained explicitly in its modules:

$$\bigcup_{i \in I} (Ex(M_i)) \subseteq Th(S).$$

CF-6 Inclusion of Complete Module Information: This is a strengthening of CF-5 to additionally include implicit information of the modules:

$$\bigcup_{i \in I} (Th(M_i)) \subseteq Th(S).$$
 For basic modules this is called local correctness in Cuenca Grau et al. (2006c), which is traced back to Garson (1989).

¹⁸In FOL with equality, the equals symbol must be noticed and can be seen as commonly shared. For example, contradictory equality sentences may produce contradictions when joining seemingly signature-disjoint theories.

- CF-7 *Deductive Conservativity*: This criterion is a kind of “inverse” to CF-6, because it requires a system to be a (deductive) conservative extension of its modules: for all $i \in I$: $Th(S)|_{Lg(M_i)} = Th(M_i)$. Put differently, every module must cover every formula of the system which can be expressed in its language. This is referred to as local completeness in Cuenca Grau et al. (2006c). The criterion of deductive conservativity is recently strongly advocated and analyzed with respect to its application in novel reasoning services for description logics (cf. among others Lutz et al., 2007a, b; Kontchakov et al., 2007; Ghilardi et al., 2006a; Cuenca Grau et al., 2008b; Kontchakov et al., 2008) for extensions of the notion of conservativity.
- CF-8 *Deductive Conservativity over Sublanguages*: A weakening of deductive conservativity yields another form of interrelating a system S and its modules M_i . Here it is not the case that every sentence of S expressible in terms of a module language $Lg(M_i)$ must be part of the module theory $Th(M_i)$, but this is only required for expressions of a sublanguage $K \subseteq Lg(S)$. One option for determining K is the use of sentence schemata. For instance, in a DL setting, Cuenca Grau et al. (2006c) defines modules to be closed under subsumption, i.e., for a sentence $\varphi \in Th(S)$ of the form $C \sqsubseteq D$ such that C or $D \in Voc(M_i)$, it is required that $\varphi \in Th(M_i)$. This definition requires atomic DL concepts to belong to one and the same module if they stand in a subsumption relation. That can produce severe effects on system structures in case of changes to the system or its modules, especially in decompositional settings if module signatures are not fixed by other means, cf. Section 3.6.3.
- CF-9 *Transitivity of Information Flow among Modules*: In general, given a system S in which the modules M_i and M_j are connected such that M_i has an impact on M_j , then composition in S should allow M_i to have an (indirect) impact on any M_k which is connected to M_j (cf. Bao et al., 2006a). Different kinds of influence lead to different variants of this criterion. One way of understanding this intuitive statement in the framework is the following. Remember that the composition operation of S should be given independently of specific arguments, i.e., in a way which can be applied to arbitrary theories (e.g., set-theoretical union). The criterion can mean that there are cases of M_i , M_j , and M_k such that $Th(M_k)$ in S differs from $Th(M_k)$ in a copy of S in which M_j ignores imports from M_i , i.e., where all (x, y) from U in S with $x \in Ifc(M_i)$ and $y \in Ifc(M_j)$ are removed.

3.5.3 Discussion of Characteristics

The above criteria have been identified as those of major relevance for the subsequent comparison of works related to modularization. The selection is further influenced by prevalent discussions in the literature. Some more advanced but less frequently stated criteria have been omitted, like “robustness under joins of signatures” (Kontchakov et al., 2007), similarly for formal, but irrelevant ones,

e.g. refraining from the inclusion of only implicit information as (an analog to the explicit case, CF-5). Hence, the list is not a complete compilation, neither with respect to the possibilities in the framework nor to the coverage of the literature.

Concerning interrelations among these characteristics, one can already notice some interplay which is likely to prevent attempts to satisfy all of them as requirements for a single notion of module. For example, it appears non-trivial to satisfy (a) directionality together with (b) the inclusion of module information in the system, (c) deductive conservativity of the system as well as (d) classically closed composition. The naïve approach to directionality fails easily, i.e., a composition operation which assigns competences to modules based on their languages. To see this in a FOL setting, let S be composed of M_1 and M_2 such that M_2 uses M_1 , and $Th(M_1) = Cn(Ex(M_1))$, $Th(M_2) = Cn(Ex(M_2) \cup Ex(M_1))$. S may answer queries in $Lg(M_1)$ only by means of the first module, thus by $Th(M_1)$, but queries in $Lg(S) \setminus Lg(M_1)$ in terms of $Th(M_2)$. This leads quickly to a situation where S loses classically closed composition, e.g., by no longer satisfying the deduction theorem of FOL.

The properties of the inclusion of module information and of deductive conservativity (CF-5 to CF-7) appear very restrictive even without the interplay with other characteristics. The question arises whether a non-trivial composition operation can be found which differs from set-theoretical union but satisfies both requirements, all the more since these characteristics originate from settings of basic modules. For the latter, black-box interfaces also exhibit some peculiarities. For modules with a single, possibly language-restricted bidirectional interface, the simple and the strict black-box variants are equivalent because of the restriction to a single source of import. Moreover, a black-box interface with output language L at a basic module M is given iff $Th^\emptyset(M)|_L = Taut(L)$. Therefore, this property corresponds to the fact that $Th(M)$ is a deductive conservative extension over every consistent interface theory $Th(F) \subseteq L$. From this follows further that linking two modules at black-box interfaces yields no interaction.

After those considerations on an abstract level, more concrete approaches can be discussed.

3.6 Analytic Overview of Logical Approaches

As stated earlier, modularization of logical theories is a rather young research interest. The following families of approaches are reviewed below.

- “conservativity and disjoint languages” (Cuenca Grau et al., 2006a, 2007a, 2008b; Ghilardi et al., 2006a, b; Kontchakov et al., 2007, 2008; Lutz et al., 2007a, b and Ponomaryov, 2006a, b, 2007, 2008)
- “partition-based reasoning” (Amir and McIlraith, 2000, 2005; MacCartney et al., 2003)
- “semantic encapsulation” (Cuenca Grau et al., 2004, 2005, 2006b, c)

- “package-based description logics” (Bao et al., 2006a, b, c, d, Cuenca Grau et al., 2007b)
- “distributed logic” (Borgida and Serafini, 2003; Bouquet et al., 2003; Ghidini and Serafini, 2000, 2001; Serafini and Bouquet, 2004, 2005)

The first four are recent approaches tackling modularization explicitly, whereas the “distributed logic” approach originates from different motivations, but applies to modularization as well. In this section, we provide a short contextual placement for each of these accounts and sketch their main ideas. Moreover, an interpretation in our framework is discussed to improve their mutual comparability, where the last subsection surveys all features in tabular form.

3.6.1 Conservativity and Disjoint Languages

The rediscovery of the logical property of conservativity and its potential use in connection with modularization issues is among the latest developments in the fields of description logics and Semantic Web ontologies. Conservativity in its deductive variant means the following (cf. also CF-7, CF-8): given two theories $T \subseteq T'$, T' is a *deductive conservative extension* of T iff T' does not entail consequences that can be expressed in the language of T , but are not already consequences of T . That means, T and T' have the same $Lg(T)$ -consequences, $Th(T)|_{Lg(T)} = Th(T')|_{Lg(T)}$. A formal-semantic and usually stronger notion is *model-conservativity*, which requires that every model of T can be extended to a model of T' without changing the interpretation of symbols in the language of T , i.e., if $M \models T$, there is an $M' \models T'$ with $M'|_{Voc(T)} = M$ (cf. also e.g. Lutz et al., 2007a; Cuenca Grau et al., 2008b) for definitional variants.

The application of conservativity to description logics is mainly studied by Frank Wolter and colleagues (Ghilardi et al., 2006a, b; Kontchakov et al., 2007; Lutz et al., 2007a, b). Their work focuses on decision and complexity problems, as well as procedures for corresponding reasoning tasks like deciding whether one DL theory is conservative over another, or computing explanations for non-conservativity among two theories. More recently, also types of conservativity and properties of such relations among theories are analyzed (Kontchakov et al., 2007; Kontchakov et al., 2008).

In terms of a more explicit approach to defining modularity for DL-based ontologies, conservativity has also been adopted as a foundation in Cuenca Grau (2007a). In this work theories are formulated in languages with their signatures being partitioned into local and external symbols. Moreover, the basic assumption is made that a modular use of the external symbols V_{ext} requires that a theory T should produce conservative extensions $T \cup T'$ over V_{ext} for every extension T' using symbols from V_{ext} . Accordingly, these theories form basic modules with bidirectional identity interfaces as they appear due to language overlap. In addition, the requirement means that the interface defined over the external signature V_{ext} is a simple black-box interface. Note that connections among two theories at these interfaces

yield no information exchange among them, cf. Section 3.5.3. Cuenca Grau et al. (2008b) is a comprehensive successor of Cuenca Grau (2007a) which adopts a weaker, relative definition of “module”: a subtheory $T' \subseteq T$ of a given DL theory T is a module of T for another DL theory S iff $S \cup T$ is a (deductively) conservative extension of $S \cup T'$ with respect to $Voc(S)$.

As a special case of conservativity one may consider the decomposition of theories into disjoint language components. The uniqueness (modulo equality formulas) of such decomposition in the first-order case has been proved by Denis Ponomaryov (Ponomaryov, 2006a, b). It appears instructive to include this case in Table 3.1 for comparison. Moreover, the approach is being developed further, recently including relative decompositions of theories, i.e., studying the relation between theories and decomposable subtheories (Ponomaryov, 2007; Ponomaryov, 2008). Accordingly, this work evolves into a partition-based reasoning approach, which is introduced next.

3.6.2 Partition-Based Reasoning

This approach is motivated with a scenario of reasoning over multiple theories with overlapping content and vocabularies in propositional or first-order logic. A subsidiary aim is the improvement of the efficiency of reasoning by means of partitions. Amir and McIlraith (2005) presents the latest introduction to the theoretical framework, which is based on earlier publications (MacCartney et al., 2003; Amir and McIlraith, 2000). The main idea is to use a message passing metaphor from the object-oriented paradigm in software engineering for reasoning over a partitioning of some theory. More precisely, given a theory T and a partitioning $(T_i)_{1 \leq i \leq n}$ of T , message passing algorithms are specified which employ “standard” reasoning within each T_i , but use message passing between certain T_i and T_k if their languages overlap. The specified algorithms are proved to be sound and complete with respect to classical reasoning over T , with these results being heavily based on Craig interpolation (Craig, 1957, Lemma 1).

Moreover, Amir and McIlraith (2005) presents a decomposition algorithm for theories which aims at minimizing three parameters, ranked by importance: for a single partition T_i , it minimizes primarily the number of symbols exchanged with other partitions, secondarily the number of unshared symbols within T_i . For the overall partitioning, the number of partitions should be kept to a minimum, which is the third and least enforced parameter.

In terms of the above framework, the theories in a partitioning are basic modules equipped with bidirectional identity interfaces over $Lg(T_i) \cap Lg(T_k)$, which are composed by set-theoretical union. Therefore, neither directionality nor deductive conservativity is satisfied in general. It remains to be studied whether the decomposition algorithm produces modules over which the system is deductively conservative. Apart from that, the decomposition approach minimizes coupling among modules, module size, and their number.

3.6.3 Semantic Encapsulation

The term “semantic encapsulation” is borrowed from Cuenca Grau (2005), and we use it to cover work on ε -connections (Kutz et al., 2004) as well as its continuation and extension especially for DLs by Bernardo Cuenca Grau and colleagues in Cuenca Grau et al. (2004, 2005, 2006b, c). ε -connections provide a method of combining certain kinds of logics, hence this approach belongs to the more mature field of combining, fibring, and fusing logics.¹⁹ There is the general result that the combination of decidable logics which can be expressed in terms of ε -connections yields a possibly more expressive, but still decidable formalism. One of the central ideas behind ε -connections is that each logic maintains its own interpretations in terms of separate model structures, but certain restrictions among the distinct model structures can be expressed when combining logics, by means of *link relations*.

Cuenca Grau et al. (2004, 2006b) apply ε -connections to the combining of ontologies based on description logics, motivated by the idea of an integrated use of independently developed OWL ontologies on the web. This work has later been extended in a way which departs from ε -connections to some extent (Cuenca Grau et al., 2006c). Nevertheless, it transfers the above-mentioned feature of the ε -connections method – distinct model structures connected by link relations – to a certain class of SHOIQ theories,²⁰ namely to such that allow for a partitioning of the domain of their models. More precisely, Cuenca Grau et al. (2006c) introduces a decomposition approach for such SHOIQ theories by presenting a partitioning algorithm for a theory T , in which the resulting set P of partitions containing subtheories is correlated with a specific group of models of T . These models exhibit a domain partitioning D , and each single partition p of T (i.e., $p \in P$) can be evaluated in a single partition $d \in D$ of the domain of the model. It is this property which may justify the name “semantic encapsulation”. Each partition has the property that any two concepts such that one entails the other belong to the same partition. Modules in the sense of Cuenca Grau et al. (2006c) are computed based on unions of such partitions, which by the computation inherit this property.²¹

With respect to our model, this account is also concerned with basic modules with identity interfaces, composed by set-theoretical union, yet in contrast to partition-based reasoning here in a DL setting. In its decompositional form, the stability of modular systems arising from such decompositions is not ensured if the initial theory is modified.

¹⁹The field emerged around the mid of the 1990s, exemplified by dedicated publications and events such as the workshop series “Frontiers of Combining Systems”; (cf. Caleiro et al., 2005).

²⁰SHOIQ is a slightly more expressive description logic than SHOIN, the description logic underlying OWL-DL. See Cuenca Grau (2005, Section 2) for an in-depth discussion.

²¹This property has an effect which may be problematic in some cases. Given a domain ontology D modularizable according to Cuenca Grau et al. (2006c), the use of an ontology with more general categories G to integrate D -categories by means of subsumption causes all modules to collapse into one. This has practically been observed in tests with the GALEN ontology (Cuenca Grau, 2005, p. 150 f.), and will prevent the use of foundational ontologies in modular fashion according to this approach.

3.6.4 Package-based Description Logics

In Bao et al. (2006d), Jie Bao and colleagues define a package-based approach for description logics, P-DL, with intended features such as a localized semantics of modules, directional semantic relations among them, and partial reuse or information hiding. Similarly to Cuenca Grau (2007a), the symbols of a theory in P-DL are divided into *foreign terms* and *home package terms*. Semantically, the disjointness of model domains as discussed above is alleviated for P-DL, i.e., importing among theories occurs by means of *domain relations*, which are specific one-to-one mappings among the domains of local models required as soon as one theory imports a foreign term.²²

In general, basic modules form the basis of this account, as well.²³ Directionality in the sense of CI-4 – though intended – is not fully supported yet, because it is possible that one module adds assumptions on foreign terms only, which propagate back to the modules these terms originate from, even if the latter do not import anything from the former (see Bao et al., 2006a, p. 627). Note further that Cuenca Grau et al. (2007b) presents a self-contained formal definition of P-DL together with a simplified, but equivalent variant which employs identity as the one-to-one mapping among model domains.

3.6.5 Distributed Logics

The notion of “distributed logics” covers several approaches rooted in the contextual reasoning community, specifically based on Local Model Semantics (LMS) and Multi-Context Systems (MCS) (cf. Ghidini and Giunchiglia, 2001; Serafini and Bouquet, 2004). More precisely, it refers to works of Luciano Serafini et al. on developing Distributed First Order Logic (DFOL) (Ghidini and Serafini, 2000), Distributed Description Logics (DDL) (Borgida and Serafini, 2003) and a contextualized form of OWL, called C-OWL (Bouquet et al., 2003).

The major idea of LMS/MCS, which conveys to the more recent approaches, is to have a collection of theories (possibly in different logical systems) each of which is first of all interpreted locally, i.e., with respect to the semantics of its associated logic (all of which are assumed to be model-theoretically defined). *Domain relations*

²²Actually, Bao et al. (2006d) discusses three types of semantics: a local semantics per module, a global semantics, in which all model domains are united and domain correspondences are merged, and a distributed semantics which includes a central package and all of its imports.

²³It is tempting to view foreign terms as defining import interfaces instead of the bidirectional identity interfaces of basic modules. This assumption is also supported by the propagation of subsumption relations among terms to modules importing all atomic term components. However, conservativity properties are not generally satisfied by P-DL, and accordingly, bidirectional interfaces appear more appropriate than import interfaces.

define (arbitrary) interconnections among the domains of such local models, which can be restricted by *compatibility constraints*, specific syntactic expressions involving different local languages. This yields information exchange among the local theories. *Bridge rules* provide a proof-theoretic counterpart for compatibility constraints: given two theories T_1 and T_2 , a bridge rule is a pair of formulas (φ, ψ) such that $\varphi \in Lg(T_1)$, $\psi \in Lg(T_2)$, which states that $\varphi \in Th(T_1)$ allows one to conclude ψ in T_2 .

The distributed logic approach is compositional rather than decompositional. Actually, it does not aim at creating an integrated system and has not been invented as an approach to tackle modularization originally. A view of many partially interrelated, coexistent theories is advocated instead. One rationale for this is the motivation to describe communicating agents with possibly different views. Despite of this, the list of properties required for DFOL (Ghidini and Serafini, 2000) documents quite some overlap with our collection of characteristics.

Local theories in DFOL can be considered as modules equipped with identity interfaces. Bridge rules provide a non-trivial composition operation, which fully supports directionality in DFOL. Further, due to the peer-to-peer semantics, the language extension at the system level is restricted to compatibility constraints. They cannot be iterated, i.e., only local formulas are permitted as subexpressions of a compatibility constraint, such that arbitrary formulas over the union of the vocabularies are not included. Put differently, the (virtual) system language is severely restricted to the union of the languages of its modules plus compatibility constraints. On a more technical level, the fact that all languages are local and mutually independent creates some inconvenience, because the representation of overlapping vocabularies implies the addition of many bridge rules.

DDL (Borgida and Serafini, 2003) transfers the approach to a description logic setting, where only restricted forms of bridge rules have been studied yet, which express forms of inter-module subsumption. However, the composition operation remains fairly unconstrained, as in the basic formalism. The effects of this freedom have been criticized not to meet expectations for modules (named the *subsumption propagation problem* and *inter-module unsatisfiability problem* (cf. Cuenca Grau et al., 2004; Bao et al., 2006a). Although these can be partially avoided by appropriate modeling (Serafini, 2005), we maintain that this mismatch originates from the motivation of linking systems with different views, which should not be intermingled with modular systems (that are assumed to be, at least potentially, globally consistent and integrated). Due to the named problems (Bao et al., 2006a) denies DDL transitive reuse, which Bao et al. define regarding subsumptions within a module. In general, however, certain effects from one module may flow to another one which is not directly connected to the first, but reachable via an intermediate module (cf. Serafini, 2005), thus satisfying transitivity of information flow as defined above (CF-9). Moreover, DDL satisfies the inclusion of implicit information (Borgida and Serafini, 2003, p. 170). Bridge rules provide directionality, except for some special cases where DDL may show effects along the reverse direction of a bridge rule (cf. Serafini, 2005).

3.6.6 Summarizing Overview

Table 3.1 shows an evaluation against the presented characteristics of modules and modular systems. The order of presentation from left to right is geared to the deviation from classical logic, rather than the chronological occurrence of the approaches. Apparently, the majority of approaches refers to basic modules composed by means of set-theoretical union. To some extent this commonality is due to our selection of works which centers on common, purely logical settings, cf. also the remarks about further relevant work in the next section. It also explains the remarkable acceptance of the inclusion of explicit module information (CF-5).

In general, the table entries provide strong indications regarding the criteria, but they should not be seen as assignments with a unique and irrevocable interpretation. For instance, in case of directionality, “No” should be read as such that it is possible to create directional compositions for basic modules as a special case, but it is not enforced in general. Similarly, compositionality with respect to consistency is only satisfied in the case of disjoint signatures if equational theories do not conflict.

Concerning modularization vs. additional motivations of the formalisms discussed, in our opinion distributed logics and P-DL intermingle different views with knowledge organization from a single point of view. We believe that these should be more clearly separated and tackled independently. Furthermore, first comparisons are available, focusing mainly on ε -connections, DDL, and P-DL. Wang et al. (2007) provides an evaluation of these approaches against a set of primarily technological criteria, clustered into five dimensions: networking, dynamics, distribution, reasoning, and expressivity. The overlap with the characteristics above is rather limited. Wang et al. (2007) follows Bao et al. (2006c) which discusses all three approaches from a DFOL perspective, showing their mutual interrelationships in this setting. In contrast, Cuenca Grau et al. (2007b) provides a comparison from a description logic point of view. In addition, the authors discuss the option to tackle modularization by novel reasoning services for existing languages, in contrast to developing languages with new, non-standard semantics. It remains open whether one of these lines offers more advantages than the other. From our perspective, reasoning services like determining the conservativity of one theory over another are definitely beneficial for managing theories. As services, however, they exhibit a dynamic character, similarly to decompositional approaches in general – which conflicts with criteria like stability (CI-2). A potential resort for this may be to distinguish “designed modularity” when building large ontologies from “dynamic modularity support” when using or analyzing ontologies.

Altogether, it seems that logical modularization currently produces diversified results and finds itself still in a phase where, based on common, high-level goals like reuse and comprehensibility, the main directions of research require further clarification.

Table 3.1 Overview of logical approaches discussed in Section 3.6 with respect to the characteristics defined in Section 3.5

Characteristic	Disjoint signatures	Basic modules with conservativity	Partition-based reasoning	Semantic encapsulation	Package-based description logic	Distributed logics
Logic	FOL	FOL, DL	FOL	DL (ε -connections)	DL	FOL, PL, DL
Way of creating modular systems	Decompositional (in Ponomaryov, 2006a, b)	Primarily compositional	Compositional and decompositional	Compositional and decompositional	Primarily compositional	Compositional
CI-1 Comprehensibility	Not addressed	Not addressed	Not addressed	Not addressed	Not addressed	Not addressed
CI-2 Stability	No	Not addressed	Rather no	No	Rather yes	Rather yes
CI-3 Compositionality wrt consistency	Yes	No	No	No	No	No
CI-4 Directionality	No	No	No	No	Partial	Yes
CF-1 Interface types	None (no interfaces)	Identity, some black-box	Identity	Identity	Identity	Identity
CF-2 Module language overlap	Only equality	Arbitrary	Arbitrary	Arbitrary	Arbitrary	None
CF-3 Classically closed composition	Yes	Yes	Yes	Yes	Yes	No
CF-4 Inclusion of explicit module information	Yes	Yes	Yes	Yes	Locally: Yes Globally: No	Yes
CF-5 Inclusion of implicit module information	Yes	Yes	Yes	Yes	Locally: Yes Globally: No	DFOL: Not appl. DDL: Yes
CF-6 Deductive conservativity	Yes	Yes	No	No	No	No
CF-7 Deductive conservativity over sublanguages	Yes (implied)	Yes (implied)	No	No	No	No
CF-8 Transitivity of information flow	No (implied)	Yes	Yes	Yes	Yes	Partially
Additional remarks		Ded. conservativity over sublanguages is discussed in Kontchakov et al. (2007)			Cf. Bao et al. (2006d) for the distinction of local and global semantics.	Restricted system language in DFOL

3.7 Concluding Remarks

3.7.1 Further Related Areas

The comparison section above concentrates on recent approaches in purely logical settings, most of which explicitly relate themselves to their application for formalizing ontologies. However, there are many other fields which may provide valuable input to theories of logical modularization.

Certainly the most prominent large-scale system involving reasoning and structuring theories has not been presented here, namely the solution provided for structuring the CYC knowledge base (Lenat and Guha, 1990) in terms of microtheories. The theory for this has primarily been developed by Rahmanathan Guha (1991), placed in the field of contextual reasoning and strongly inspired by John McCarthy, who later continued work on contexts with a related formalism (McCarthy and Buvač, 1998). Obrst (2010, Section 4) in this volume summarizes both microtheories and reasoning about contexts. Contextual reasoning is at the borderline of our selection. We just note that covering this approach as well appears possible and would clearly transcend basic modules, e.g., by the need for actual transformations in interfaces.

In the Semantic Web area there is a lot of research which either extends logical formalisms more radically, adds extra-logical features, or may also provide less logic-oriented approaches, e.g. for decomposing ontologies in RDF(S) based on properties of RDF graphs. Examples for this overall group are Stuckenschmidt and Klein (2003, 2004) and Seidenberg and Rector (2006).

Another very relevant line of research refers to formal approaches in software engineering and in the semantics of programs and programming languages. cursorily, a number of authors including ourselves already draw inspiration upon software engineering, e.g. by referring to notions like interfaces, information hiding, etc. However, established formal results in these fields should be studied more closely, e.g., approaches such as Bergstra (1990) and Diaconescu et al. (1993). The latter is founded on category theory (cf. Adámek et al., 1990; Healy, 2010, this volume). On a category-theoretic basis several related approaches have been developed. One of them involves the Common Algebraic Specification Language (CASL) (Mosses, 2004), which was originally designed for algebraic software specification. CASL and its extension HetCASL (Mossakowski, 2005), respectively, allow for logical specifications in a variety of systems. They are recently employed for ontologies (Lüttich et al., 2006) by means of the Heterogeneous Toolset (Hets) (Mossakowski, 2005). Concerning modularity, that represents an interesting approach insofar as concepts for structuring in CASL are only very loosely dependent on the specific logic in use. Kutz and Mossakowski (2008) discusses theoretical foundations for this direction, suggesting the category-theoretic notions of diagram, colimits, and of the institutions of Goguen and Burstall (1992) as a proper basis for modularity in ontologies. This approach appears promising, despite and due to its high

level of abstraction; e.g. (Kutz and Mossakowski, 2008) can directly align itself with the conservativity approaches in Section 3.6, re-interprets ε -connections (Kutz et al., 2004) in the semantic encapsulation approach and DDL (Borgida and Serafini, 2003) in the distributed logics family, and supports implementations with Hets (Mossakowski, 2005).

Finally, the Information Flow Framework (IFF)²⁴ developed by Robert Kent (2005) remains to be included. On a strict category-theoretic foundation Kent defines a metatheory intended to serve for the structuring of the Standard Upper Ontology²⁵ (see also Obrst, 2010, Section 4.5, this volume). Similarly to HetCASL and Kutz and Mossakowski (2008), a major aim of the IFF is to achieve independence of particular logics in which ontologies may be expressed. Accordingly, both approaches are also closely related to the field of combining logics, while the motivation of the IFF is intimately tied to the organization of ontologies.

3.7.2 Conclusions

First, let us briefly summarize this chapter. Tracing the route of terminologies and ontologies in information systems, we outline the move of (medical) terminological systems from simple term lists to formal description logic theories. From this perspective, reducing the complexity of systems was first tackled by composing complex concepts from atomic concepts, as well as by a structured arrangement of concepts. Ensuring correct concept composition has ultimately been achieved by the use of description logics, which can be used to prove formal consistency and to avoid meaningless concepts to some extent. Nevertheless, the problems of how to comprehensibly arrange, search, and navigate large structures of categories like polyhierarchies or graphs remain open, despite recent research and some advances.

Due to the status of (description) logics as the contemporary formalism to express ontologies, e.g. in the medical domain or the Semantic Web, we concentrate on logical modularization approaches as a means to tackle the complexity of large categorical systems to the extent of facilitating their comprehensibility, construction, maintainability, evolution and reuse. Regarding work concentrating on ontologies, a corresponding field is currently being established, with a major focus on description logics for the Semantic Web. In order to compare its approaches and discussions, we present an abstract framework to describe characteristics of modules and related notions. On the one hand, this framework is tuned to a high level of generality with sufficient degrees of freedom in order to cover many proposals. On the other hand, it attempts to remain conceptually minimal, by a model established on top of theories mainly by the notions of interface, module, and system. Basically, modules are understood as components of systems which are interconnected at

²⁴Information Flow Framework: <http://suo.ieee.org/IFF/>

²⁵Standard Upper Ontology: <http://suo.ieee.org/>

interfaces, exchanging logical sentences. Notably, there are model-based approaches to modularization which seem to deviate from the idea of exchanging sentences.

In the previous section, a number of approaches are introduced and considered regarding this framework. That section indicates the fairly heterogeneous landscape of modularization proposals in terms of the characteristics introduced in Section 3.5. Other characteristics were not covered, like the application of the approaches for top-level ontologies as discussed in Loebe (2006), or more generally the utility and adequacy with respect to specialized purposes. It remains to be seen how those proposals behave along other dimensions.

It may appear questionable to discuss modularization for logical systems rather than genuinely for ontologies. However, we are not aware of any particular ontological theory of modularization, nor elaborate methodological guidance which concerns the structuring of categorical systems. Currently, we see Rector (2003) as the closest work to attempting such guidance. In connection with logical approaches it is noteworthy that the core of all approaches covers a partitioning/division of the logical *languages*. Ontologically, this may be interpreted to refer to different domains, which leads one to the theory of levels of reality (Gnoli and Poli, 2004) and of domains in general. Those may become an initial step to ontological principles of organization. However, this requires further studies, see also chapters Herre (2010b) in this volume and Symons (2010) in previous volume. To provide an example, a distinction with logical impact and potential benefit by means of an ontological approach refers to special cases of partitions of categories. In Loebe (2006), taxonomic interrelations of such partitions are distinguished: *horizontal* partitions comprising categories on a similar level of generality, yet in different domains, which are contrasted with *vertical* partitions organized in such a way that all categories in one partition are more general than one or more members in another. Cuenca Grau et al. (2006a) adheres to a similar difference and defines diverse characteristics for modularization in correspondence with these cases. Pursuing such routes further may transcend purely formal solutions, possibly founded on more elaborate ontological theories about categories and domains than available at present.

Acknowledgements We would like to thank Heinrich Herre, Frank Wolter, Denis Ponomaryov, Alexander Heußner, and Roberto Poli for valuable comments on earlier versions of the manuscript, and we gratefully acknowledge the members of the research group Onto-Med at the University of Leipzig for maintaining a lively and inspiring atmosphere.

References

- Adámek, J., H. Herrlich, and G.E. Strecker. 1990. *Abstract and concrete categories: The joy of cats*. Vol. in Pure and applied mathematics. New York: John Wiley & Sons. Improved online edition (2004): <http://katmat.math.uni-bremen.de/acc>
- Amir, E., and S. McIlraith. 2000. Partition-based logical reasoning. In *Principles of knowledge representation and reasoning*, Proceedings of the Seventh International Conference (KR 2000), Breckenridge, CO, 11–15 Apr 2000, eds. A.G. Cohn, F. Giunchiglia, and B. Selman, 389–400. San Francisco, CA: Morgan Kaufmann.
- Amir, E., and S. McIlraith. 2005. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence* 162(1–2):49–88.

- Ashburner, M., C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, and J.T. Eppig, et al. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics* 25(1):25–29.
- Baader, F. 2003. Terminological cycles in a description logic with existential restrictions. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, 9–15 Aug 2003, 325–330. San Francisco, CA: Morgan Kaufmann.
- Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider eds. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.
- Baader, F., S. Brandt, and C. Lutz. 2005. Pushing the EL envelope. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, Scotland, 30 Jul–5 Aug 2005, 364–369. San Francisco, CA: Morgan Kaufmann.
- Baader, F., C. Lutz, and B. Suntisrivaraporn. 2006. Efficient reasoning in EL+. In Proceedings of the 2006 International Workshop on Description Logics (DL 2006), Lake District, UK, 30 May–1 Jun 2006, CEUR 189, 15–26.
- Bao, J., D. Caragea, and V.G. Honavar. 2006a. Modular ontologies – A formal investigation of semantics and expressivity. In *The semantic web – ASWC 2006*, Proceedings of the First Asian Semantic Web Conference, Beijing, China, 3–7 Sept 2006, LNCS 4185, eds. R. Mizoguchi, Z. Shi, and F. Giunchiglia, 616–631. Berlin: Springer.
- Bao, J., D. Caragea, and V.G. Honavar. 2006b. On the semantics of linking and importing in modular ontologies. In Proceedings of the 5th International Semantic Web Conference, Athens, GA, 5–9 Nov 2006, LNCS 4273, eds. I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, 72–86. Berlin: Springer.
- Bao, J., D. Caragea, and V.G. Honavar. 2006c. On the semantics of linking and importing in modular ontologies. Computer Science Technical Reports Nr. 465, Department of Computer Science, Iowa State University, Ames, IA.
- Bao, J., D. Caragea, and V.G. Honavar. 2006d. Towards collaborative environments for ontology construction and sharing. In Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS 2006), Las Vegas, NV, 14–17 May 2006, 99–108. Piscataway, NJ: IEEE Press.
- Barwise, J., and S. Feferman eds. 1985. *Model-theoretic logics*. Vol. in Perspectives in mathematical logic. Berlin: Springer.
- Bergstra, J.A., J. Heering, and P. Klint. 1990. Module algebra. *Journal of the ACM* 37(2):335–372.
- Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The semantic web. *Scientific American* 284(5):34–43.
- Borgida, A., and L. Serafini. 2003. Distributed description logics: assimilating information from peer sources. *Journal on Data Semantics* 1:153–184.
- Bouquet, P., F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. 2003. C-OWL: Contextualizing ontologies. In *The semantic web – ISWC 2003*, Proceedings of the Second International Semantic Web Conference, Sanibel Island, FL, 20–23 Oct. LNCS 2870, eds. D. Fensel, K. Sycara, and J. Mylopoulos, 164–179. Berlin: Springer.
- Brandt, S. 2004. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – What else? In *Frontiers in artificial intelligence and engineering*, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, 22–27 Aug 2004, eds. R.L. de Mántaras, and L. Saitta, Vol. 110 in Frontiers in Artificial Intelligence and Engineering, 298–302. Amsterdam: IOS Press.
- Caleiro, C., A. Sernadas, C. Sernadas. 2005. Fibring logics: Past, present and future. In *We will show them: Essays in honour of Dov Gabbay*, eds. S. Artemov, H. Barringer, A.S. d’Avila Garcez, L.C. Lamb, and J. Woods, Vol. 1, 363–388. London: King’s College Publications.
- Chang, C.C., and H.J. Keisler. 1990. *Model theory*. Vol. 73 in Studies in Logic and the Foundations of Mathematics, 3rd ed. Amsterdam: Elsevier.
- Cimino, J.J. 1998. Desiderata for controlled medical vocabularies in the twenty-first century. *Methods of Information in Medicine* 37(4/5):394–403.

- Craig, W. 1957. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic* 22(3):269–285.
- Cruz, I., S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo eds. 2006. *The semantic web – ISWC 2006*, Proceedings of the 5th International Semantic Web Conference, Athens, GA, 5–9 Nov 2006, LNCS 4273. Berlin: Springer.
- Cuenca Grau, B., B. Parsia, and E. Sirin. 2004. Working with multiple ontologies on the semantic web. In Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 7–11 Nov 2004, LNCS 3298, eds. S.A. McIlraith, D. Plexousakis, and F. van Harmelen 620–634. Berlin: Springer.
- Cuenca Grau, B. 2005. Combination and integration of semantic web ontologies. Ph.D. thesis, Department of Informatics, Valencia University, Valencia, Spain.
- Cuenca Grau, B., I. Horrocks, O. Kutz, and U. Sattler. 2006. Will my ontologies fit together? In Proceedings of the 2006 International Workshop on Description Logics (DL 2006), Lake District, UK, 30 May–1 Jun 2006, CEUR 189, eds. B. Parsia, U. Sattler, and D. Toman 175–182. Aachen, Germany: CEUR-WS.org
- Cuenca Grau, B., B. Parsia, and E. Sirin. 2006. Combining OWL ontologies using ϵ -Connections. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 4(1): 40–59.
- Cuenca Grau, B., B. Parsia, E. Sirin, and A. Kalyanpur. 2006. Modularity and web ontologies. In Proceedings of the Tenth International Conference (KR 2006), Lake District, UK, 2–5 Jun, 2006, 198–209.
- Cuenca Grau, B., I. Horrocks, Y. Kazakov, and U. Sattler. 2007. A logical framework for modularity of ontologies. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, 6–12 Jan, 2007, 298–303. San Francisco, CA: Morgan Kaufmann.
- Cuenca Grau, B., O. Kutz, and V. Honavar. 2007. Modular ontology languages revisited. In Proceedings of the Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa) held at IJCAI 2007, Hyderabad, India, 7 Jan, eds. T. Finin, D. Caragea, D. Mladenic, and Y. Sure.
- Cuenca Grau, B., V. Honavar, A. Schlicht, and F. Wolter eds. 2008a. Proceedings of the 2nd International Workshop on Modular Ontologies (WoMO 2007), Whistler, Canada, 28 Oct 2008, CEUR 315. Aachen, Germany: CEUR-WS.org.
- Cuenca Grau, B., I. Horrocks, Y. Kazakov, and U. Sattler. 2008b. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research* 31:273–318.
- Diaconescu, R., J. Goguen, and P. Stefaneas. 1993. Logical support for modularisation. In *Logical Environments*, eds. G. Huet, and G. Plotkin, ch. 4, 83–130. New York: Cambridge University Press. Proceedings of the Second Annual Workshop on Logical Environments, Edinburgh, Scotland.
- Doherty, P., J. Mylopoulos, and C.A. Welty eds. 2006. *Principles of Knowledge Representation and Reasoning*, Proceedings of the Tenth International Conference (KR 2006), Lake District, UK, 2–5 Jun, 2006. Menlo Park, CA: AAAI Press.
- Dzbor, M., and E. Motta. 2008. Engineering and customizing ontologies: the human-computer challenge in ontology engineering. In *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*, Vol. 7 in Semantic Web and Beyond: Computing for Human Experience, eds. M. Hepp, P. De Leenheer, A. de Moor, and Y. Sure, 25–57. Berlin: Springer.
- Ebbinghaus, HD., J. Flum, and W. Thomas. 1994. *Mathematical logic*. Vol. in Undergraduate Texts in Mathematics, 2nd ed. Berlin: Springer.
- Euzenat, J., and P. Shvaiko. 2007. *Handbook of ontology matching*. Berlin: Springer.
- Forrey, A.W., C.J. McDonald, G. DeMoor, S.M. Huff, D. Leavelle, D. Leland, T. Fiers, L. Charles, B. Griffin, F. Stalling, A. Tullis, K. Hutchins, and J. Baenzinger. 1996. Logical observation identifier names and codes (LOINC) database: a public use set of codes and names for electronic reporting of clinical laboratory test results. *Clinical Chemistry* 42(1):81–90.

- Gangemi, A., D.M. Pisanelli, and G. Steve. 1999. An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data & Knowledge Engineering* 31(2):183–220.
- Garson, J. 1989. Modularity and relevant logic. *Notre Dame Journal of Formal Logic* 30(2): 207–223.
- Ghidini, C., and L. Serafini. 2000. Distributed first order logic. In *Frontiers of Combining Systems 2*, eds. D. Gabbay, and M. de Rijke, 121–139. No. 7 in Studies in Logic and Computation. Papers presented at FroCoS'98, Amsterdam, 1998, Baldock (UK): Research Studies Press Ltd.
- Ghidini, C., and F. Giunchiglia. 2001. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence* 127(2):221–259.
- Ghilardi, S., C. Lutz, and F. Wolter. 2006. Did I damage my ontology: A case for conservative extensions of description logics. In *Principles of Knowledge Representation and Reasoning. Proceedings of the Tenth International Conference (KR 2006)*, Lake District, UK, 2–5 Jun, 2006, eds. P. Doherty, J. Mylopoulos, and C.A. Welty, 187–197. San Francisco, CA: Morgan Kaufmann.
- Ghilardi, S., C. Lutz, F. Wolter, and M. Zakharyashev. 2006. Conservative extensions in modal logic. In *Advances in Modal Logic*, Volume 6 (AiML-2006), Noosa, Queensland, Australia, 25–28 Sep, 2006, eds. G. Governatori, I.M. Hodkinson, and Y. Venema, 187–207. College Publications.
- Gnoli, C., and R. Poli. 2004. Levels of reality and levels of representation. *Knowledge Organization* 31(3):151–160.
- Goguen, J., and R. Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM* 39(1):95–146.
- Gómez-Pérez, A., M. Fernández-López, and O. Corcho. 2004. *Ontological engineering: with examples from the areas of knowledge management, e-commerce and the semantic web*. Vol. in Advanced Information and Knowledge Processing. London: Springer.
- Gottlob, G., and T. Walsh. eds. 2003. Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, 9–15 Aug 2003. San Francisco, CA: Morgan Kaufmann.
- Guha, R.V. 1991. Contexts: A formalization and some applications. Ph.D Thesis/Technical Report STAN-CS-91-1399, Computer Science Department, Stanford University.
- Haase, P., V.G. Honavar, O. Kutz, Y. Sure, and A. Taminin eds. 2006. Proceedings of the First International Workshop on Modular Ontologies, (WoMO 2006), Athens, GA, 5 Nov 2006, CEUR 232. Aachen, Germany: CEUR-WS.org.
- Healy, M. 2010. Ontology and category theory. In *TAO – Theory and Applications of Ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 3, ch. 3.
- Hearst, M.A. 2006. Design recommendations for hierarchical faceted search interfaces. In Proceedings of the First Workshop on Faceted Search held at SIGIR 2006, Seattle, WA, Aug 10. eds. A.Z. Broder, and Y.S. Maarek, [Paper available from:] <http://flamenco.berkeley.edu/papers/faceted-workshop06.pdf>
- Hendler, J., and F. van Harmelen. 2007. The semantic web: webizing knowledge representation. In *Handbook of Knowledge Representation, Foundations of Artificial Intelligence*, eds. F. van Harmelen, V. Lifschitz, and B. Porter, ch. 21, 821–839. Amsterdam: Elsevier.
- Herre, H. 1995. Generalized compactness of nonmonotonic inference operations. *Journal of Applied Non-Classical Logics* 5(1):121–135.
- Herre, H., B. Heller, P. Burek, R. Hoehndorf, F. Loebe, and H. Michalek. 2006. General Formal Ontology (GFO) – A foundational ontology integrating objects and processes [Version 1.0]. Onto-Med Report 8, Research Group Ontologies in Medicine, Institute of Medical Informatics, Statistics and Epidemiology, University of Leipzig, Leipzig, Germany.
- Herre, H. 2010a. General Formal Ontology: A foundational ontology for conceptual modeling. In *TAO – Theory and Applications of Ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 2, ch. 4.
- Herre, H. 2010b. The ontology of medical terminological systems: Towards the next generation of biomedical ontologies. In *TAO – Theory and Applications of Ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 3, ch. 1.

- Hildebrand, M., J. van Ossenbruggen, and L. Hardman. 2006. /facet: A browser for heterogeneous semantic web repositories. In Proceedings of the 5th International Semantic Web Conference, Athens, GA, 5–9 Nov 2006, LNCS 4273, eds. I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, 272–285.
- Iggulden, P., and C. Price. 2001. SNOMED clinical terms – Looking forward to the next phase. *British Journal of Healthcare Computing and Information Management* 18(10):20–22.
- Kalfoglou, Y., and M. Schorlemmer. 2003. Ontology mapping: the state of the art. *The Knowledge Engineering Review* 18(1):1–31.
- Kalfoglou, Y., and M. Schorlemmer. 2010. Ontology alignment. In *TAO – Theory and Applications of Ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 1, ch. 6.
- de Keizer, N.F., and A. Abu-Hanna. 2000. Understanding terminological systems II: Experience with conceptual and formal representation of structure. *Methods of Information in Medicine* 39(1):22–29.
- de Keizer, N.F., A. Abu-Hanna, and J.H.M. Zwetsloot-Schonk. 2000. Understanding terminological systems I: Terminology and typology. *Methods of Information in Medicine* 39(1):16–21.
- Kelso, J., K. Prüfer, and R. Hoehndorf. 2010. Ontologies in biology. In *TAO – Theory and Applications of Ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 3, ch. 2.
- Kent, R.E. 2005. Semantic integration in the Information Flow Framework. In *Semantic Interoperability and Integration*, eds. Y. Kalfoglou, W.M. Schorlemmer, A.P. Sheth, S. Staab, and M. Uschold, Dagstuhl Seminar Proceedings 04391. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.
- Kontchakov, R., F. Wolter, and M. Zakharyashev. 2007. Modularity in DL-Lite. In Proceedings of the 20th International Workshop on Description Logics (DL-2007), Brixen-Bressanone, Italy, 8–10 Jun. CEUR 250, eds. D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A. Turhan, and S. Tessaris, 76–87. Aachen, Germany: CEUR-WS.org.
- Kontchakov, R., F. Wolter, and M. Zakharyashev. 2008. Can you tell the difference between DL-Lite ontologies? In *Principles of Knowledge Representation and Reasoning*, eds. G. Brewka, and J. Lang, 285–295. Proceedings of the 11th International Conference (KR 2008), Sydney, Australia, 16–19 Sept. Menlo Park, CA: AAAI Press.
- Krisnadhi, A., and C. Lutz. 2007. Data complexity in the EL family of description logics. In *Logic for Programming, Artificial Intelligence, and Reasoning*, eds. N. Dershowitz, and A. Voronkov, 333–347. Proceedings of the 14th International Conference (LPAR 2007), Yerevan, Armenia, 15–19 Oct 2007. LNCS 4790. Berlin: Springer.
- Kutz, O., C. Lutz, F. Wolter, and M. Zakharyashev. 2004. ε -Connections of abstract description systems. *Artificial Intelligence* 156(1):1–73.
- Kutz, O., and T. Mossakowski. 2008. Modules in transition: conservativity, composition, and colimits. In Proceedings of the 2nd International Workshop on Modular Ontologies (WoMO 2007), Whistler, Canada, 28 Oct 2008.
- Lenat, D.B., and R.V. Guha. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading, MA: Addison-Wesley.
- Loebe, F. 2006. Requirements for logical modules. In Proceedings of the First International Workshop on Modular Ontologies, (WoMO 2006), Athens, GA, 5 Nov 2006, CEUR 232, eds. Haase, P., V.G. Honavar, O. Kutz, Y. Sure, and A. Taminlin. Aachen, Germany: CEUR-WS.org
- Lüttich, K., C. Masolo, and S. Borgo. 2006. Development of modular ontologies in CASL. In Proceedings of the First International Workshop on Modular Ontologies, (WoMO 2006), Athens, GA, 5 Nov 2006, CEUR 232, eds. Haase, P., V.G. Honavar, O. Kutz, Y. Sure, and A. Taminlin. Aachen, Germany: CEUR-WS.org
- Lutz, C., D. Walther, and F. Wolter. 2007. Conservative extensions in expressive description logics. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, 6–12 Jan, 2007, 453–458.

- Lutz, C., and F. Wolter. 2007. Conservative extensions in the lightweight description logic EL. In *Automated Deduction – CADE-21*, ed. F. Pfenning, 84–99. Berlin: Springer. Proceedings of the 21st International Conference on Automated Deduction, Bremen, Germany, 17–20 Jul. LNCS 4603.
- MacCartney, B., S.A. McIlraith, E. Amir, and T.E. Uribe. 2003. Practical partition-based theorem proving for large knowledge bases. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, 9–15 Aug 2003, 89–98.
- McCarthy, J., and S. Buvač. 1998. Formalizing context (expanded notes). In *Computing Natural Language, Vol. 81 of CSLI Lecture Notes*, eds. A. Aliseda, R.J. van Glabbeek, and D. Westerståhl, 13–50. Center for the Study of Language and Information (CSLI), Stanford University.
- McDonald, C.J., S.M. Huff, J.G. Suico, G. Hill, D. Leavelle, R. Aller, A. Forrey, K. Mercer, G. DeMoor, J. Hook, W. Williams, J. Case, and P. Maloney. 2003. LOINC, a universal standard for identifying laboratory observations: A 5-year update. *Clinical Chemistry* 49(4):624–633.
- McIlraith, S.A., D. Plexousakis, and F. van Harmelen eds. 2004. *The semantic web – ISWC 2004: Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan, 7–11 Nov 2004, LNCS 3298. Berlin: Springer.
- Mosses, P.D. ed. 2004. *CASL reference manual: The complete documentation of the common algebraic specification language*. LNCS 2960. Berlin: Springer.
- Mossakowski, T. 2005. Heterogeneous specification and the heterogeneous tool set. Habilitation Thesis. University of Bremen, Germany.
- Noy, N.F. 2004. Semantic integration: a survey of ontology-based approaches. *SIGMOD Record* 33(4):65–70.
- Orbst, L. 2010. Ontological architectures In *TAO – Theory and applications of ontology*, eds. M. Healy, A. Kameas, and R. Poli, Vol. 2: The information science stance. Part 1, ch. 2. Heidelberg: Springer.
- Pack Kaelbling, L., and A. Saffiotti eds. 2005. Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, Scotland, 30 Jul–5 Aug 2005. Denver, CO: Professional Book Center.
- Pan, Z., A. Qasem, and J. Heflin. 2006. An investigation into the feasibility of the semantic web. In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006) and the 18th Innovative Applications of Artificial Intelligence Conference (IAAI 2006), Boston, MA, 16–20 July 2006, 1394–1399. Menlo Park, CA: AAAI Press.
- Parsia, B., U. Sattler, and D. Toman eds. 2006. Proceedings of the 2006 International Workshop on Description Logics (DL 2006), Lake District, UK, 30 May–1 Jun 2006, CEUR 189. Aachen, Germany: CEUR-WS.org.
- Ponamaryov, D. 2006. Formal knowledge representation and the decomposability problem. Technical Report 135 (translated version), Institute of Informatics Systems, Siberian Division of the Russian Academy of Sciences, Novosibirsk, Russia.
- Ponamaryov, D. 2006. Semantic web basics in logical consideration. In eds. P. Hitzler, and Y. Sure, Proceedings of the First International Workshop on Applications of Semantic Technologies (AST 2006), Dresden, Germany, 6 Oct 2006.
- Ponamaryov, D. 2007. Generalized decomposability notions for first-order theories. *Bulletin of the Novosibirsk Computing Center* 26:103–110.
- Ponamaryov, D. 2008. A decomposability criterion for elementary theories. *Siberian Mathematical Journal* 49(1):152–154.
- Priss, U. 2000. Faceted knowledge representation. *Electronic Transactions on Artificial Intelligence* 4(Section C):21–33.
- Ranganathan, S.R. 1962. *Elements of library classification*. 3rd ed. Bombay: Asia Publishing House.
- Ranganathan, S.R. 1967. *Prolegomena of library classification*. Bombay: Asia Publishing House.

- Rector, A.L., S. Bechhofer, C.A. Goble, I. Horrocks, W.A. Nowlan, and W.D. Solomon. 1996. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9(2):139–171.
- Rector, A.L. 1999. Clinical terminology: why is it so hard? *Methods of Information in Medicine* 38(4/5):239–252.
- Rector, A.L. 2003. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In Proceedings of the Second International Conference on Knowledge Capture (K-CAP 2003), Sanibel Island, FL, 23–25 Oct 2003, eds. J. Gennari, B. Porter, and Y. Gil, 121–128. New York: ACM Press.
- Rector, A.L., and J. Rogers. 2005. Ontological & practical issues in using a description logic to represent medical concepts: experience from GALEN. Preprint Series CSPP-35, School of Computer Science, The University of Manchester.
- Rogers, J., A. Roberts, D. Solomon, E. van der Haring, C. Wroe, P. Zanstra, and A.L. Rector. 2001. GALEN ten years on: Tasks and supporting tools. In *MedInfo 2001: Towards Global Health: The Informatics Route to Knowledge*. Proceedings of the Tenth World Congress on Health and Medical Informatics of the International Medical Informatics Association, London, 2–5 Sep 2001. Vol. 84 of Studies in Health Technology and Informatics, eds. V. Patel, R. Rogers, and R. Haux, 256–260. Amsterdam: IOS Press.
- Rossi Mori, A. 1997. A second generation of terminological systems is coming. In Proceedings of the 13th Medical Informatics Europe (MIE 1997), Porto Carras, Greece, 25–29 May 1997. Vol. 43 of Studies in Health Technology and Informatics, ed. C. Pappas, 436–440. Amsterdam: IOS Press.
- Sacco, G.M. 2000. Dynamic taxonomies: A model for large information bases. *IEEE Transactions on Knowledge and Data Engineering* 12(3):468–479.
- Sacco, G.M. 2006. Some research results in dynamic taxonomy and faceted search systems. In eds. A.Z. Broder, and Y.S. Maarek, Proceedings of the First Workshop on Faceted Search held at SIGIR 2006, Seattle, WA, 10 Aug.
- Sattler, U. and A. Tamilin eds. 2008. Proceedings of the Workshop on Ontologies: Reasoning and Modularity (WORM 2008), Tenerife, Spain, 2 Jun. CEUR 348. Aachen, Germany: CEUR-WS.org.
- Seidenberg, J., and A. Rector. 2006. Web ontology segmentation: Analysis, classification and use. In Proceedings of the 15th International Conference on World Wide Web (WWW 2006), Edinburgh, Scotland, 23–26 May 2006, eds. L. Carr, D. De Roure, A. Iyengar, C. Goble, and M. Dahlin, 13–22. New York: ACM Press.
- Serafini, L., and P. Bouquet. 2004. Comparing formal theories of context in AI. *Artificial Intelligence* 155(1–2):41–67.
- Serafini, L., A. Borgida, and A. Tamilin. 2005. Aspects of Distributed and Modular Ontology Reasoning. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, 570–575.
- Sowa, J.F. 2000. *Knowledge representation: Logical, philosophical and computational foundations*. Pacific Grove: Brooks/Cole.
- Spackman, K.A., and K.E. Campbell. 1998. Compositional concept representation using SNOMED: Towards further convergence of clinical terminologies. In *A paradigm shift in health care information systems: Clinical infrastructures for the 21st century*, Proceedings of the 1998 AMIA Annual Symposium, Orlando, FL, 7–11 Nov 1998, ed. C.G. Chute, 740–744. Philadelphia, PA: Hanley and Belfus.
- Spackman, K.A. 2001. Normal forms for description logic expressions of clinical concepts in SNOMED RT. In *A medical informatics odyssey: Visions of the future and lessons from the past*, Proceedings of the 25th AMIA Annual Symposium 2001, Washington, DC, 3–7 Nov 2001, ed. S. Bakken, 627–631. Bethesda, MD: American Medical Informatics Association.
- Specia, L., and E. Motta. 2007. Integrating folksonomies with the semantic web. In *The Semantic Web: Research and Applications*, Proceedings of the 4th European Semantic Web Conference

- (ESWC 2007), Innsbruck, Austria, 3–7 Jun 2007, LNCS 4519, eds. E. Franconi, M. Kifer, and W. May, 624–639. Berlin: Springer.
- Spiteri, L. 1998. A simplified model for facet analysis: Ranganathan 101. *Canadian Journal of Information and Library Science* 23(1–2):1–30.
- Straub, H.R. 2002. Four different types of classification models. In *Knowledge media in healthcare: Opportunities and challenges*, ed. R. Grütter, 57–82. Hershey, PA: Idea Group Publishing.
- Stuckenschmidt, H., and M. Klein. 2003. Modularization of ontologies. Deliverable D21, IST Project 2001–33052 Wonderweb, Vrije Universiteit, Amsterdam.
- Stuckenschmidt, H., and M. Klein. 2004. Structure-based partitioning of large concept hierarchies. In *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan, 7–11 Nov 2004, LNCS 3298, 289–303. Berlin: Springer.
- Symons, J. 2010. Levels of reality In *TAO – Theory and applications of ontology*, eds. R. Poli, J. Seibt, and J. Symons, Vol. 1: The philosophical stance. Part 1, ch. 6.
- Veloso, M.M. ed. 2007. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, Hyderabad, India, 6–12 Jan, 2007. Menlo Park, CA: IJCAI.
- W3C. 2004a. Resource description framework (RDF) and RDF schema (RDFS) specifications. World Wide Web Consortium (W3C). <http://www.w3.org/RDF/>
- W3C. 2004b. Web ontology language (OWL) specifications. World Wide Web Consortium (W3C). <http://www.w3.org/2004/OWL/>
- Wang, Y., J. Bao, P. Haase, and G. Qi. 2007. Evaluating formalisms for modular ontologies in distributed information systems. In *Web reasoning and rule systems*, Proceedings of the First International Conference (RR 2007), Innsbruck, Austria, 7–8 Jun 2007, LNCS 4524, eds. M. Marchiori, J.Z. Pan, C. de Sainte Marie, 178–193. Berlin: Springer.
- WHO. 2004. History of the development of the ICD. World Health Organization (WHO). <http://www.who.int/classifications/icd/en/HistoryOfICD.pdf>
- Wright, S.E., G. Budin eds. 1997. *Handbook of terminology management*. Amsterdam: John Benjamins.

Chapter 4

The Information Flow Approach to Ontology-Based Semantic Alignment

Yannis Kalfoglou and Marco Schorlemmer

4.1 Introduction

In order for two systems (databases, software agents, peers, web services, software components, etc.) to be considered semantically integrated, both will need to commit to a shared conceptualisation of the application domain. Commonly, this is achieved by providing an explicit specification of this conceptualisation – what has become to be known as an *ontology* – and by defining each system’s local vocabulary in terms of the ontology’s vocabulary. Thus, an ontology models the vocabulary used by knowledge engineers so that it denotes concepts and their relations, and it constrains the interpretation of this vocabulary to the meaning originally intended by knowledge engineers. As such, ontologies have been widely adopted as an enabling technology for interoperability in distributed environments, such as multi-agent systems, federated databases, or the semantic web.

This sort of interoperability is dubbed “semantic” precisely because it assumes that the ontology is some sort of structured theory T – coming thus equipped with a precise semantics for the structure it holds – and because each system’s local language L_i is *interpreted* in T (e.g., in the technical sense of a theory interpretation as defined in (Enderton, 2002), when T is a theory in first-order logic). Semantic integration is therefore always relative to the theory T into which local languages are interpreted. We shall call this theory the *reference theory* of the integration.

The use of ontologies as reference theories for semantic integration, however, is more in tune with a classical codification-centred knowledge management tradition, as put forward in (Corrêa da Silva and Agustí, 2003). Such tradition comprises the efforts to define standard upper-level ontologies such as CyC (Lenat, 1995) and SUO (IEEE, 2003), or to establish public ontology repositories for specific domains to favour knowledge reuse such as the Ontolingua server (Farquhar et al., 1997). Corrêa da Silva and Agustí remark that “centralised ontologies [...] promise to

Y. Kalfoglou (✉)
Ricoh Europe Plc, London, UK
e-mail: ykalfoglou@googlemail.com

bring the control of the organisation back to what was possible under classical management techniques. The problem is that they may also bring back the rigidity of agencies organised under the classical management tenets.”

Before ontologies became popular, knowledge engineers hardly ever had to work with more than one ontology at a time. Even in cases where multiple ontologies were used (see, e.g., Borst et al., 1997), these were mostly controlled experiments (e.g., Uschold et al., 1998) in moderated environments (such as Farquhar et al., 1997). Nowadays, however, the practice is somewhat different. Modern trends in knowledge management dictate that we should expect to work more and more within highly distributed, open, and dynamic environments like the web. In this sort of environment it is more realistic to achieve certain levels of semantic integration by matching vocabulary on-the-fly. In addition, the proliferation of many diverse ontologies caused by different conceptualisations of even the same domain – and their subsequent specification using varying terminology – has highlighted the need of ontology matching techniques that are capable of computing semantic relationships between entities of disparate ontologies (Kalfoglou and Schorlemmer, 2003b; Shvaiko and Euzenat, 2005). Since ontologies are the result of an inter-subjective agreement among individuals about the same fragment of the objective world, they are also highly context-dependent and hardly will result to be general-purpose, regardless of how abstract and upper-level they might be.

4.2 Ontology-Based Semantic Integration: Basic Concepts and Definitions

In this chapter we shall be concerned with semantic integration understood as the integration of two systems by virtue of the interpretation of their respective vocabularies into a reference theory – an ontology – expressible in some logical language. In practice, semantic integration is often carried out on subsets of first-order logic, such as description logics (DL), for which reasoning has good computational properties. This is, for instance, the approach followed by Calvanese and De Giacomo in their ontology integration system for database schemata (Calvanese and De Giacomo, 2005); W3C, too, has embraced DLs in order to develop the OWL recommendation for ontology representation (McGuinness and van Harmelen, 2004). Another example is the focus of Giunchiglia, Marchese and Zaihrayeu on propositional DLs in order to use fast SAT provers for matching taxonomically organised vocabularies (Giunchiglia et al., 2006). In contrast, the Process Specification Language (PSL) is an example of a semantic integration initiative based on full first-order logic that uses invariants to define interpretations of local vocabulary into PSL (Grüninger and Kopena, 2005).

By *vocabulary* we mean a set V of words and symbols used by a system to represent and organise its local knowledge. In a formal, logic-based representation language the vocabulary is constituted by the non-logical symbols used to form sentences and formulae (in this case it is usually referred to as *parameters* or

signature). The *language* is then the set $L(V)$ of all well-formed formulae over a given vocabulary V . We shall also write L when we do not want to explicitly refer to the vocabulary. We call the elements of a language L , *sentences*.

In declarative representation languages, knowledge is represented and organised by means of theories. DL-based ontologies are such an example. A convenient way to abstractly characterise theories in general, is by means of a consequence relation. Given a language L , a *consequence relation* over L is, in general, a binary relation $|-$ on subsets of L which satisfies certain structural properties.¹ Consequence relations are also suitable to capture other sorts of mathematical structures used to organise knowledge in a systematic way, such as taxonomic hierarchies. When defined as a binary relation on L (and not on subsets of L), for instance, it coincides with a partial order. Furthermore, there exists a close relationship between consequence and classification relations (which play a central role in ontological knowledge organisation), which has been thoroughly studied from a mathematical perspective in (Dunn and Hardegree, 2001; Barwise and Seligman, 1997; Ganter and Wille, 1999).

We call a *theory* a tuple $T = \langle L_T, |-_T \rangle$, where $|-_T \subseteq \wp(L_T) \times \wp(L_T)$ is a consequence relation, hence capturing with this notion the formal structure of an ontology in general. Finally, in order to capture the relationship between theories, we call a *theory interpretation* a map between the underlying languages of theories that respects consequence relations. That is, a function $i: L_T \rightarrow L_{T'}$ is a theory interpretation between theories $T = \langle L_T, |-_T \rangle$ and $T' = \langle L_{T'}, |-_{T'} \rangle$ if, and only if, for all $\Gamma, \Delta \subseteq L$ we have that $\Gamma |-_T \Delta$ implies $i(\Gamma) |-_{T'} i(\Delta)$ (where $i(\Gamma)$ and $i(\Delta)$ are the set of direct images of Γ and Δ along i , respectively).²

4.2.1 Semantic Matching

We call *semantic matching* the process that takes two theories T_1 and T_2 as input (called *local theories*) and computes a third theory $T_{1 \leftrightarrow 2}$ as output (called *bridge theory*) that captures the semantic relationship between T_1 and T_2 's languages with respect to a reference theory T . As we shall see below, we call the output of the semantic-matching process, together with the input it relates, a *semantic alignment*. It is important to make a couple of remarks here.

First, one usually distinguishes a theory from its presentation. If the language L is infinite (as for instance in propositional or first-order languages, where the set of well-formed formulae is infinite, despite having a finite vocabulary), any consequence relations over L will also be infinite. Therefore, one deals in practice with a finite subset of $\wp(L) \times \wp(L)$, called a *presentation*, to stand for the smallest consequence relation containing this subset. A presentation may be empty, in which case the smallest consequence relation over a language L containing it, is called the

¹These are commonly those of Identity, Weakening and Global Cut (see Definition 9).

²Theories and theory interpretations as treated here can also be seen as particular cases of the more general framework provided by institution theory, which has been thoroughly studied in the field of algebraic software specification (see Goguen and Burstall, 1992).

trivial theory. We will write $Tr(L)$ for the trivial theory over L . It is easy to prove that, for all $\Gamma, \Delta \subseteq L$, $\Gamma \dashv_{Tr(L)} \Delta$ if, and only if, $\Gamma \cap \Delta \neq \emptyset$.

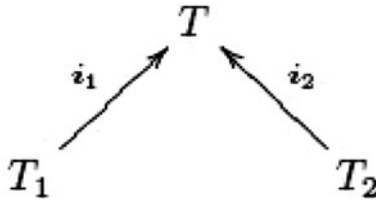
Rigorously speaking, current implementations of semantic matching actually take two presentations of local theories as input and compute a presentation of the bridge theory as output. But, from a conceptual perspective, we shall characterise semantic matching always in terms of the theories themselves.

Second, the reference theory T is usually *not* an explicit input to the semantic matching process (not even a presentation of it). Instead it should be understood as the background knowledge used by a semantic matcher to infer semantic relationships between the underlying languages of the respective input theories. For a manual matcher, for instance, the reference theory may be entirely dependent on user input, while a fully automatic matcher would need to rely on automatic services (either internal or external to the matcher) to infer such reference theory. It is for this reason that we talk of a *virtual* reference theory, since it is not explicitly provided to the semantic matcher, but is implicit in the way external and internal sources are brought into the matching process as background theory in order to compute a semantic alignment.

Next, we provide precise definitions of what we mean by bridge theory to capture a semantic alignment of languages, and also what we mean by a semantic alignment underlying a semantic integration of local theories.

4.2.2 Integration Theory

Definition 1: Two theories T_1 and T_2 are *semantically integrated with respect to T* , if there exist theory interpretations $i_1 : T_1 \rightarrow T$ and $i_2 : T_2 \rightarrow T$.



We call $\mathbb{I} = \{i_j : T_j \rightarrow T\}_{j=1,2}$ the *semantic integration* of local theories T_1 and T_2 with respect to *reference theory T* . Two languages L_1 and L_2 are *semantically integrated with respect to T* if their respective trivial theories are.

In a semantic alignment we are interested in determining the semantic relationship between the languages L_{T_1} and L_{T_2} on which semantically integrated theories T_1 and T_2 are expressed. Therefore, a semantic integration \mathbb{I} of T_1 and T_2 with respect to a reference theory T as defined above is not of direct use, yet. What we would like to have is a theory $T_{\mathbb{I}}$ over the combined language $L_{T_1} \uplus L_{T_2}$ (the disjoint union) expressing the semantic relationship that arises by interpreting local theories in T . We call this the *integration theory* of \mathbb{I} , and it is defined as the inverse image of the reference theory T under the sum of the theory interpretations in \mathbb{I} .

Definition 2: Let $i : T \rightarrow T'$ be a theory interpretation. The *inverse image* of T' under i , denoted $i^{-1}[T']$, is the theory over the language of T such that $\Gamma \upharpoonright_{i^{-1}[T]} \Delta$ if, and only if, $i(\Gamma) \upharpoonright_{-T'} i(\Delta)$.

It is easy to prove that, for every theory interpretation $i : T \rightarrow T'$, T is a *subtheory* of $i^{-1}[T']$, i.e., $|-_T \subseteq |-_{i^{-1}[T']}$

Definition 3: Given theories $T_1 = \langle L_{T_1}, |-_{T_1} \rangle$ and $T_2 = \langle L_{T_2}, |-_{T_2} \rangle$, the sum $T_1 + T_2$ of theories is the theory over the sum of language (i.e., the disjoint union of languages) $L_{T_1} \uplus L_{T_2}$ such that $|-_{T_1+T_2}$ is the smallest consequence relation such that $|-_{T_1} \subseteq |-_{T_1+T_2}$ and $|-_{T_2} \subseteq |-_{T_1+T_2}$.

Given theory interpretations $i_1 : T_1 \rightarrow T$ and $i_2 : T_2 \rightarrow T$, the sum $i_1 + i_2 : T_1 \rightarrow T_2 \rightarrow T$ of theory interpretations is just the sum of their underlying map of languages.

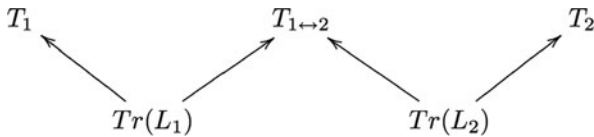
Definition 4: Let $I = \{i_j : T_j \rightarrow T\}_{j=1,2}$ be a semantic integration of T_1 and T_2 with respect to T . The *integration theory* T_{\perp} of the semantic integration \mathbb{I} is the inverse image of T under the sum of interpretations $i_1 + i_2$, i.e. $T_{\perp} = (i_1 + i_2)^{-1}[T]$.

The integration theory faithfully captures the semantic relationships between sentences in L_{T_1} and L_{T_2} as determined by their respective interpretation into T , but expressed as a theory over the combined language $L_{T_1} \uplus L_{T_2}$. The sum of local theories $T_1 + T_2$ is therefore always a subtheory of the integration theory T_{\perp} , because it is through the interpretations in T where we get the semantic relationship between languages. It captures and formalises the intuitive idea that an integration is more than just the sum of its parts.

4.2.3 Semantic Alignment

In semantic matching one usually isolates as output to the matching process the bit that makes T_{\perp} genuinely a super theory of $T_1 + T_2$. The idea is to characterise a theory $T_{1 \leftrightarrow 2}$ over the disjoint union of subsets $L_1 \subseteq L_{T_1}$ and $L_2 \subseteq L_{T_2}$, called bridge theory, which, together with T_1 and T_2 , uniquely determines the integration theory T_{\perp} . To keep everything characterised uniformly in the same conceptual framework, the bridge theory, together with its relationship to the local theories T_1 and T_2 , can be expressed by a diagram of theory interpretations as follows.

Definition 5: A *semantic alignment* \mathcal{A} of T_1 with T_2 is a diagram



in the category of theories and theory interpretations, where $L_i \subseteq L_{T_i}$ and $T_{1 \leftrightarrow 2}$ is a theory whose underlying language $L_{T_{1 \leftrightarrow 2}} = L_1 \uplus L_2$, and where all arrows are theory inclusions. We shall also write $T_1 \xleftarrow{\mathcal{A}} T_2$ as shorthand of an alignment.

We say that a semantic alignment A underlies a semantic integration I when the colimit of A in the category of theories and theory interpretations (which always exists) is the integration theory of \mathbb{T} , i.e., $\text{colim}(A) = T_{\mathbb{T}}$.

This representation of semantic alignment as a system of objects and morphisms in a category, and of semantic integration by means of a colimit of such a diagram, bears a close relationship to the notion of *W-alignment diagram* described in (Zimmermann et al., 2006). This is so because both notions share the same categorical approach to semantic alignment. But, unlike in (Zimmermann et al., 2006), we further take a dual “type-token” structure of semantic integration into account, and we define an alignment with respect to this two-tier model. We claim that in this way we better capture Barwise and Seligman’s basic insight that “information flow involves both types and their particulars” (Barwise and Seligman, 1997). This will become clearer next when we describe the role of tokens in semantic alignment scenarios.

4.3 Semantic Alignment Through Meaning Coordination

We shall consider a scenario in which two agents A_1 and A_2 want to interoperate, but each agent A_i has its knowledge represented according to its own conceptualisation, which we assume is explicitly specified by means of its own ontology O_i . Any expression α_i using the vocabulary O_i will be considered semantically distinct a priori from any expression α_j using vocabulary O_j (with $j \neq i$), even if they happen to be syntactically equal, unless the semantic evidence unveiled by an ontology-matching process of the kind described below makes them mean the same to A_1 and A_2 . Furthermore, we assume that the agents’ ontologies are not open for inspection, so that semantic heterogeneity cannot be solved by semantically matching the ontologies beforehand.

An agent may learn about the ontology of another agent only through meaning coordination. Thus, we assume that agent A_i is capable of requesting from agent A_j to explain the intended meaning of an expression α_j that is in a message from A_j to A_i and uses the vocabulary O_j . Agent A_i might request such an explanation with the intention of determining the semantic relationship of the fragment of O_j used in α_j with respect to its local vocabulary O_i . Correspondingly, we assume that agent A_j is capable of explaining to A_i the meaning of expression α_j by means of a *token* of this expression.

The formal framework we describe in the next section is neutral with respect to the syntactic form of expressions and, more importantly, to what tokens might be, giving an interesting level of generality to ontology alignment. The Oxford Dictionary of English defines a token as “a thing serving as a visible or tangible representation of something abstract.” In our scenario a token will be something agent A_i is capable of processing and putting into relationship with its own local ontology O_i .

Take for instance the ontology negotiation process described in (Bailin and Truszkowski, 2002). There, an agent A_i , upon the reception from another agent

A_j of a message containing a list of keywords, either sends to A_j an interpretation of the keywords in the form of WordNet synonyms in order to check that it has interpreted A_j 's vocabulary correctly, or else requests A_j for a clarification of the interpretation of unknown keywords, also in form of WordNet synonyms. Thus, in this scenario, the role of tokens is played by WordNet synonyms of those keywords whose interpretation needs to be confirmed or clarified.

Looking at another ontology alignment scenario (Wang and Gasser, 2002) present an ontology-matching algorithm for open multi-agent systems, where ontologies are partitions of domain instances into categories, based on the K-means algorithm, a typical partition-based clustering method. The alignment is computed out in an online fashion by exchanging instances between two agents, rather than by exchanging abstract concepts. When an agent plans to express some concept or category to other agents it uses an instance belonging to that category to represent this concept. In this scenario it is particular domain instances who play the role of tokens of a concept or category. Wang and Gasser further note, that “unless a set of agents already has a compatible and verified shared ontology, it is difficult to see how they could specify categories to each other in another way.” The capability of a set of agents to process and classify tokens according to their own local ontologies is what underlies the ontology-matching process. van Diggelen et al. (2007) also describe an ontology matching protocol pointing to instances for concept explication. One agent communicates a number of positive and negative examples of the concept to the other agent, which in turn, classifies these examples using the concept classifier from its own ontology.

Finally, in other scenarios (Giunchiglia and Shvaiko, 2004) and (Bouquet et al., 2003) use mappings of concepts in a tree hierarchy to propositional expressions using WordNet synsets in order to check, by means of a SAT prover (a software program that checks the satisfiability of the propositions supplied to it), the semantic relationships between concepts occurring in two different hierarchies. In this scenario, a concept is represented by a propositional formula, playing the role of the token for this concept, which can then be processed by each agent with the SAT prover.

4.4 Semantic Alignment Hypotheses

We have described a process by which agents compute an ontology alignment by making the intended meaning of syntactic expressions explicit to each other through the use of tokens for these expressions. We deliberately have left unspecified what these tokens actually are, and have only briefly mentioned that we shall consider tokens as something agents are capable of processing and putting into relationship with their own local vocabulary. This view of a semantic alignment is the result of the research initiated by (Kent, 2000) on conceptual knowledge organization, and applied to ontology alignment by (Schorlemmer and Kalfoglou, 2003; Kalfoglou and Schorlemmer, 2004) aiming at a formal foundation for semantic interoperability and integration based on channel theory – Barwise and Seligman's proposal for a mathematical theory of information (Barwise and Seligman, 1997).

In this section we introduce the main channel-theoretic constructs required for our formal foundation for ontology alignment, motivating them by means of three *Semantic Alignment Hypotheses*.

Channel theory takes the idea of a classification as the fundamental notion for modelling the local context by which tokens relate to types:

Definition 6: A *classification* $\mathbf{A} = \langle tok(\mathbf{A}), typ(\mathbf{A}), \models_{\mathbf{A}} \rangle$ consists of a set of *tokens* $tok(\mathbf{A})$, a set of *types* $typ(\mathbf{A})$ and a *classification relation* $\models_{\mathbf{A}} \subseteq tok(\mathbf{A}) \times typ(\mathbf{A})$ that classifies tokens to types.

Although a very simple notion, classifications have recently been used, under varying terminology, in many related fields of formal knowledge representation and theoretical computer science (e.g., in algebraic logic (Dunn and Hardegree, 2001), categorical logic (Barr, 1996), formal concept analysis (Ganter and Wille, 1999), and process algebra (Pratt, 2001)).

Hypothesis 2: *Semantic alignment presupposes a flow of information between expressions (i.e., types) of separate agents that happens by virtue of shared tokens for these expressions. This flow of information can be accurately described by means of an information channel (Definition 8).*

A fundamental construct of channel theory is that of an information channel between two classifications. It models the information flow between components. First, though, we need to describe how classifications are connected with each other through infomorphisms:

Definition 7: An *infomorphism* $f = \langle f^{\rightarrow}, f^{\leftarrow} \rangle: \mathbf{A} \rightarrow \mathbf{B}$ from classifications \mathbf{A} to \mathbf{B} is a contravariant pair of functions $f^{\rightarrow}: typ(\mathbf{A}) \rightarrow typ(\mathbf{B})$ and $f^{\leftarrow}: tok(\mathbf{B}) \rightarrow tok(\mathbf{A})$ satisfying the following fundamental property, for each type $\alpha \in typ(\mathbf{A})$ and token $b \in tok(\mathbf{B})$:

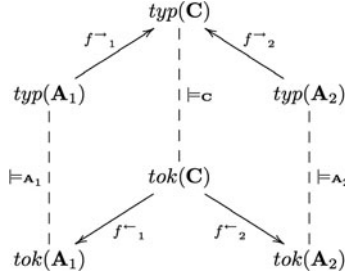
$$f^{\leftarrow}(b) \models_{\mathbf{A}} \alpha \quad \text{iff} \quad b \models_{\mathbf{B}} f^{\rightarrow}(\alpha)$$

$$\begin{array}{ccc}
 \alpha & \xrightarrow{f^{\rightarrow}} & f^{\rightarrow}(\alpha) \\
 \models_{\mathbf{A}} \downarrow & & \downarrow \models_{\mathbf{B}} \\
 f^{\leftarrow}(b) & \xleftarrow{f^{\leftarrow}} & b
 \end{array}$$

As with classifications, infomorphisms have been around in the literature for a long time, and its contra-variance between the type- and token- level is recurrent in many fields. They would correspond to *interpretations* when translating between

logical languages (Enderton, 2002), or to *Chu transforms* in the context of *Chu spaces* (Pratt, 1995). Channel theory makes use of this contra variance to model the flow of information at type-level because of the particular connections that happen at the token-level:

Definition 8: An *information channel* consists of two classifications \mathbf{A}_1 and \mathbf{A}_2 connected through a core classification \mathbf{C} via two infomorphisms f_1 and f_2 :



Hypothesis 3: *Semantic alignment is formally characterised by a consequence relation between expressions (i.e., types) of separate agents. This consequence relation can be faithfully captured by the natural logic (Definition 11) of the core of the information channel underlying the integration.*

Channel theory is based on the understanding that information flow is the result of regularities in distributed systems. These regularities are implicit in the representation of systems as interconnected classifications. However, one can make these regularities explicit in a logical fashion by means of theories and local logics:

Definition 9: A *theory* $T = \langle typ(T), \vdash_T \rangle$ consists of a set $typ(T)$ of types, and a binary relation between subsets of $typ(T)$. Pairs $\langle \Gamma, \Delta \rangle$ of subsets of $typ(T)$ are called *sequents*. If $\Gamma \vdash_T \Delta$, for $\Gamma, \Delta \subseteq typ(T)$, then the sequent $\Gamma \vdash_T \Delta$ is called a *constraint*. T is regular if for all $\alpha \in typ(T)$ and all $\Gamma, \Gamma', \Delta, \Delta', \Sigma \subseteq typ(T)$:

1. *Identity:* $\alpha \vdash_T \alpha$
2. *Weakening:* If $\Gamma \vdash_T \Delta$, then $\Gamma, \Gamma' \vdash_T \Delta, \Delta'$
3. *Global Cut:* If $\Gamma, \Sigma_0 \vdash_T \Delta, \Sigma_1$ for each partition $\langle \Sigma_0, \Sigma_1 \rangle$ of Σ , then $\Gamma \vdash_T \Delta$

Note that, as is usual with sequents and constraints, we write α instead of $\{\alpha\}$ and Γ, Γ' instead of $\Gamma \cup \Gamma'$. Also, a partition of Σ is a pair $\langle \Sigma_0, \Sigma_1 \rangle$ of subsets of Σ , such that $\Sigma_0 \cup \Sigma_1 = \Sigma$ and $\Sigma_0 \cap \Sigma_1 = \emptyset$; Σ_0 and Σ_1 may themselves be empty (hence it is actually a quasi-partition). Note that Global Cut is implied by the

usual (Finitary) Cut only if the binary relation is *compact*, i.e., $\Gamma \upharpoonright_{-T} \Delta$ implies the existence of finite subsets $\Gamma_0 \subseteq \Gamma$ and $\Delta_0 \subseteq \Delta$ such that $\Gamma_0 \upharpoonright_{-T} \Delta_0$.

Regularity arises from the observation that, given any classification of tokens to types, the set of all sequents that are satisfied by all tokens always fulfills Identity, Weakening, and Global Cut. Hence, the notion of a local logic:

Definition 10: A *local logic* $\mathbb{L} = \langle tok(\mathbb{L}), typ(\mathbb{L}), |=_{\mathbb{L}}, \upharpoonright_{-\mathbb{L}}, N_{\mathbb{L}} \rangle$ consists of a classification $cla(\mathbb{L}) = \langle tok(\mathbb{L}), typ(\mathbb{L}), |=_{\mathbb{L}} \rangle$, a regular theory $th(\mathbb{L}) = \langle typ(\mathbb{L}), \upharpoonright_{-\mathbb{L}} \rangle$ and a subset of $N_{\mathbb{L}} \subseteq tok(\mathbb{L})$ of *normal tokens*, which satisfy all the constraints of $th(\mathbb{L})$; a token $a \in tok(\mathbb{L})$ satisfies a constraint $\Gamma \upharpoonright_{-\mathbb{L}} \Delta$ of $th(\mathbb{L})$ if, when a is of all types in Γ , a is of some type in Δ .

Finally, every classification determines a natural logic, which captures the regularities of the classification in a logical fashion, and which we shall use in order model the semantic interoperability between agents with different ontologies:

Definition 11: The *natural logic* is the local logic $Log(\mathbf{C})$ generated from a classification \mathbf{C} , and has as classification \mathbf{C} , as regular theory the theory whose constraints are the sequents satisfied by all tokens, and whose tokens are all normal.

The three Semantic Alignment Hypotheses above comprise the core of what we call *the information-flow approach to ontology-based semantic alignment*. The basic concepts and definitions of Section 4.2 characterise semantic alignment in terms of theory interpretations, which amount to maps of languages, actually maps of types. Hypotheses 1 and 2, however, make the role of tokens explicit in the characterisation of a semantic integration. The natural logic then determines the integration theory of Section 4.2 entirely through the way tokens are classified to types in the core of an information channel, thus playing the role of the reference theory of the integration. In the next section we summarise how we have been applying this view of semantic integration in order to successfully tackle the semantic heterogeneity problem in a variety of different scenarios.

4.5 Applications and Explorations

Ontology Mapping: A thorough survey on existing ontology mapping techniques in this domain revealed a surprising scarcity of formal, theoretically sound approaches to the problem (Kalfoglou and Schorlemmer, 2003b). Consequently, we set out to explore information-flow theoretic ways to tackle the problem. In (Kalfoglou and Schorlemmer, 2003a) we describe a novel ontology mapping method and a system that implements it, IF-Map, which aims to (semi-)automatically map ontologies by representing ontologies as IF classifications and automatically generate infomorphisms between them. We demonstrated this approach by using the IF-Map system to map ontologies in the domain of computer science departments from five UK universities. The underlying philosophy of IF-Map follows the assumption that the way communities classify their instances

with respect to local types reveals the semantics that could be used to guide the mapping process. The method is operationalised in a system that includes harvesting mechanisms for acquiring ontologies from online resources, translators for processing different ontology representation formalisms, and APIs for web-enabled access of the generated mappings, all in the form of infomorphisms which are encoded in RDF/OWL formats.

Theory of Semantic Interoperability: We have also explored the suitability of the information flow theory to define a framework that captures semantic interoperability without committing to any particular semantic perspective (model-theoretic, property-theoretic, proof-theoretic, etc.), but which accommodates different understandings of semantics (Kalfoglou and Schorlemmer, 2004). We articulated this framework around four steps that, starting from a characterisation of an interoperability scenario in terms of IF classifications of tokens to types, define an information channel that faithfully captures the scenario's semantic interoperability. We used this framework in an e-government alignment scenario, where we used our four-step methodology to align UK and US Governmental departments using their ministerial units as types and their respective set of responsibilities as tokens, which were classified against those types.

Progressive Ontology Alignment: More recently, we applied information-flow theory to address the issues arising during ontology coordination (Schorlemmer and Kalfoglou, 2004; Schorlemmer and Kalfoglou, 2005). We have been modelling ontology coordination with the concept of a coordinated information channel, which is an IF channel that states how ontologies are progressively coordinated, and which represents the semantic integration achieved through interaction between two agents. It is a mathematical model of ontology coordination that captures the degree of participation of an agent at any stage of the coordination process, and is determined both, at the type and at the token level. Although not yet a fully-fledged theory of ontology coordination, nor an ontology coordination methodology or procedure, we have illustrated our ideas in a scenario taken from (Sowa, 2000) where one needs to coordinate different conceptualisations in the English and French language of the concepts of "river" and "stream" on one side, and "fleuve" and "reivière" on the other side.

Situated Semantic Alignment: Most ontology matching mechanisms developed so far have taken a classical functional approach to the semantic heterogeneity problem, in which ontology matching is seen as a process taking two or more ontologies as input and producing a semantic alignment of ontological entities as output (Giunchiglia and Shvaiko, 2004). Furthermore, matching often has been carried out at design-time, before integrating knowledge-based systems or making them interoperate. But, multi-agent communication, peer-to-peer information sharing, and web-service composition are all of a decentralised, dynamic, and open-ended nature, and they require ontology matching to be locally performed during run-time. In addition, in many situations peer ontologies are not even open for inspection (e.g., when they are based on commercially confidential information). Atencia and Schorlemmer (2007) claim that a semantic alignment of ontological terminology is ultimately relative to the particular situation in which the alignment is

computed, and that this situation should be made explicit and brought into the alignment mechanism. Even two agents with identical conceptualisation capabilities, and using exactly the same vocabulary to specify their respective conceptualisations, may fail to interoperate in a concrete situation because of their differing perception of the domain. They address the case in which agents are already endowed with a top-down engineered ontology (it can even be the same one), which they do not adapt or refine, but for which they want to find the semantic relationships with separate ontologies of other agents on the grounds of their communication within a specific situation. In particular, they provide a formal model that formalises situated semantic alignment as a sequence of information-channel refinements capturing the flow of information occurring in distributed systems due to the particular situations – or tokens – that carry information. Analogously, the semantic alignment that will allow information to flow ultimately will be carried by the particular situation agents are acting in (Atencia and Schorlemmer, 2008).

4.6 Conclusions

We have approached the limits of ontology-based semantic alignment from its mathematical foundations and in the context of alignment scenarios in open and distributed environments, like the Web, and its extension, the Semantic Web. We argued for the need to address what we believe is still a lack of sound mathematical models of information, semantics, and interoperability for multi-agent systems, and distributed knowledge models on the Semantic Web (Kalfoglou et al., 2004). We showed that we needed to go beyond the usual approach, which models semantic alignment as the first-order interpretation of dissimilar vocabularies into a common ontology.

We propose a general theory of semantic integration that uses a logic-independent formulation of language, ontology, and ontological commitment that can cope with the variety of logics and understandings of semantics occurring in highly decentralised and distributed environments. Furthermore, our proposed theory defines semantic alignment on top of this logic-independent formulation by means of channel theory. In particular we have shown that the natural logic of the core of an information channel adequately and faithfully captures the intuitive consequence relation lying behind semantically aligned systems. This led us to advocate for a channel-theoretic characterisation of semantic alignment that we stated in the form of three *Semantic Alignment Hypotheses*. Such channel-theoretic characterisation allowed us to look beyond the standard ontology-based approach to semantic alignment, and we illustrated this by means of interaction-based meaning coordination between agents.

By providing a sound theoretical ground upon which we base our three hypotheses for enabling semantic alignment, we enable the use of our framework to model semantic-alignment as it occurs in semantic heterogeneity scenarios by applying a variety of technologies. Instead of exploring concrete instantiations of the formal model to particular alignment technologies – wandering into the discussion of particular choice methods, termination criteria, and alignment algorithms – we decided

to shift our attention to what basic capability an agent should have to be able to engage in an ontology-alignment interaction. Choice of tokens and types, interaction termination criteria, and concrete matching algorithms will play a central role when grounding the formal model in concrete domains. This has been explored in two exemplar uses of our work: progressive ontology alignment and situated semantic alignment.

Acknowledgements This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01; under Agreement Technologies CONSOLIDER-INGENIO 2010 (CSD2007-00022) and MULOG2 (TIN2007-68005-C04-01) by Spain's Ministerio de Ciencia e Innovación; and by the Generalitat de Catalunya (2009-SGR-1434).

References

- Atencia, M., and M. Schorlemmer. 2007. A formal model for situated semantic alignment. In eds. E.H. Durfee, and M. Yokoo, *Proceedings of the Sixth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007)*, 1270–1277. New York, NY: ACM.
- Atencia, M., and M. Schorlemmer. 2008. I-SSA: Interaction-situated semantic alignment. On the move to meaningful Internet systems: OTM 2008. OTM 2008 Confederated International Conferences. CoopIS, DOA, GADA, IS, and ODBASE 2008. Monterrey, Mexico, November 9–14, 2008. *Proceedings, Part I*, volume 5331 of *Lecture Notes in Computer Science*, 445–455. Berlin: Springer.
- Bailin, S., and W. Truszkowski. 2002. Ontology negotiation between intelligent information agents. *The Knowledge Engineering Review* 17(1):7–19.
- Barr, M. 1996. The Chu construction. *Theory and Applications of Categories* 2(2):17–35.
- Barwise, J., and J. Seligman. 1997. Information flow: The logic of distributed systems, volume 44 of *Cambridge tracts in theoretical computer science*. Cambridge: Cambridge University Press.
- Borst, P., H. Akkermans, and J. Top. 1997. Engineering ontologies. *International Journal of Human-Computer Studies* 46(2–3):365–406.
- Bouquet, P., F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. 2003. C-OWL: Contextualizing ontologies. In *The semantic web – ISWC 2003, volume 2870 of Lecture Notes in Computer Science*, eds. D. Fensel et al., 164–179. Heidelberg: Springer.
- Calvanese, D., and G. De Giacomo. 2005. Data integration: A logic-based perspective. *AI Magazine* 26(1):59–70.
- Corrêa da Silva, F., and J. Agusti. 2003. *Knowledge coordination*. New York: Wiley.
- Dunn, J.M., and G.M. Hardegree. 2001. *Algebraic methods in philosophical logic*. Oxford: Oxford University Press.
- Enderton, H. 2002. *A mathematical introduction to logic*, 2nd edn. San Diego: Academic Press.
- Farquhar, A., R. Fikes, and J. Rice. 1997. The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies* 46(6):707–727.
- Ganter, B., and R. Wille. 1999. *Formal Concept Analysis*. Berlin: Springer.
- Giunchiglia, F., M. Marchese, and I. Zaihrayeu. 2006. Encoding classifications into lightweight ontologies. In *The semantic web: Research and applications, volume 4011 of Lecture Notes in Computer Science*, eds. Y. Sure, and J. Domingue, 80–94. Berlin: Springer.
- Giunchiglia, F., and P. Shvaiko. 2004. Semantic matching. *The Knowledge Engineering Review* 18(3):265–280.
- Goguen, J., and R. Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM* 39(1):95–146.
- Grüninger, M. and J.B. Kopena. 2005. Semantic integration through invariants. *AI Magazine* 26(1):11–20.

- IEEE 18 Dec 2003. Standard Upper Ontology Working Group (SUO WG). <http://suo.ieee.org>. Last retrieved on 15 Sept 2007.
- Kalfoglou, Y., H. Alani, M. Schorlemmer, and C. Walton. 2004. On the emergent semantic web and overlooked issues. In *The semantic web – ISWC 2004, volume 3298 of Lecture Notes in Computer Science*, eds. S. McIlraith, et al., 576–590. Berlin: Springer.
- Kalfoglou, Y., and M. Schorlemmer. 2003a. IF-Map: An ontology-mapping method based on information-flow theory. In *Journal on data semantics I*, volume 2800 of *Lecture Notes in Computer Science*, eds. S. Spaccapietra, et al., 98–127. Heidelberg: Springer.
- Kalfoglou, Y., and M. Schorlemmer. 2003b. Ontology mapping: The state of the art. *The Knowledge Engineering Review* 18(1):1–31.
- Kalfoglou, Y., and M. Schorlemmer. 2004. Formal support for representing and automating semantic inter-operability. In *The semantic web: Research and applications, volume 3053 of Lecture Notes in Computer Science*, eds. C. Bussler, et al., 45–60. Springer.
- Kent, R.E. 2000. The information flow foundation for conceptual knowledge organization. In *Proceedings of the Sixth International Conference of the International Society for Knowledge Organization*.
- Lenat, D. 1995. CyC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.
- McGuinness, D., and F. van Harmelen. 2004. OWL web ontology language overview. Technical report, World Wide Web Consortium (W3C). Last retrieved on 15 Sept 2008, <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.
- Pratt, V. 1995. The Stone gamut: A coordinatization of mathematics. In *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, 444–454. IEEE Computer Society Press.
- Pratt, V. 2001. Orthocurrence as both interaction and observation. In eds. R. Rodriguez, and F. Anger, *Proceedings of the IJCAI'01 Workshop on Spatial and Temporal Reasoning*.
- Schorlemmer, M., and Y. Kalfoglou. 2003. On semantic interoperability and the flow of information. In *Semantic Integration*, eds. A. Doan, A. Halevy, and N. Noy, volume 82 of *CEUR Workshop Proceedings*.
- Schorlemmer, M., and Y. Kalfoglou. 2004. A Channel-theoretic foundation for ontology coordination. In *Proceedings of the ISWC'04 Workshop on Meaning Coordination and Negotiation*.
- Schorlemmer, M., and Y. Kalfoglou. 2005. Progressive ontology alignment for meaning coordination: An information-theoretic foundation. In eds. F. Dignum, et al., *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 737–744. New York, NY: ACM Press.
- Shvaiko, P., and J. Euzenat. 2005. A survey of schema-based matching approaches. In *Journal on Data Semantics IV, volume 3730 of Lecture Notes in Computer Science*, eds. S. Spaccapietra et al., 146–171. Berlin: Springer.
- Sowa, J.F. 2000. *Knowledge representation: logical, philosophical, and computational foundations*. Pacific Grove, CA: Brooks/Cole.
- Uschold, M., M. Healy, K. Williamson, P. Clark, and S. Woods. 1998. Ontology reuse and application. In *Formal ontology in information systems, volume 46 of Frontiers in Artificial Intelligence and Applications*, ed. N. Guarino, 179–192. Amsterdam: IOS Press.
- van Diggelen, J., R.-J. Beun, F. Dignum, R.M. van Eijk, and J.-J. Meyer. 2007. Ontology negotiation: goals, requirements and implementation. *International Journal of Agent-Oriented Software Engineering* 1(1):63–90.
- Wang, J., and L. Gasser. 2002. Mutual online ontology alignment. In *OAS'02 Ontologies in Agent Systems, Proceedings of the AAMAS 2002 Workshop*, volume 66 of *CEUR Workshop Proceedings*.
- Zimmermann, A., M. Krötzsch, J. Euzenat, and P. Hitzler. 2006. Formalizing ontology alignment and its operations with category theory. In *Formal Ontology in Information Systems*, eds. B. Bennett, and C. Fellbaum, *Proceedings of the Fourth International Conference (FOIS 2006)*, volume 150 of *Frontiers in Artificial Intelligence and Applications*. Amsterdam: IOS Press.

Chapter 5

Ontological Evaluation and Validation

Samir Tartir, I. Budak Arpinar, and Amit P. Sheth

5.1 Introduction

Building an ontology for a specific domain can start from scratch (Cristani and Cuel, 2005) or by modifying an existing ontology (Gómez-Pérez and Rojas-Amaya, 1999). In both cases, techniques for evaluating the characteristics and the validity of the ontology are necessary. Not only such techniques might be useful during the ontology engineering process (Paslaru et al., 2006), they can also be useful to an end-user who is looking for an ontology that is suitable for her application domain. The user can select the best ontology according to her application needs among several ontologies (Sabou et al., 2005).

Ontology evaluation is an important task that is needed in many situations. For example, during the process of building of an ontology, ontology evaluation is important to guarantee that what is built meets the application requirement. Fernández et al. (1999) presents a life cycle for ontologies (Fig. 5.1). The life cycle is mainly based on Software Engineering processes. Their cycle includes three sets of activities: Management (that includes control and quality control), technical (that includes tasks for building an ontology), and support (that includes activities that are performed at the same time as the technical tasks). In this methodology, ontology evaluation was presented as an ongoing process throughout the ontology lifecycle in both the management and the support activities to illustrate its importance. Ontology evaluation is also important in cases where the ontology is automatically populated from different resources that might not be homogeneous, leading to duplicate instances, or instances that are clustered according to their sources in the same ontology, both of which may decrease the usefulness of the ontology. For example, the search for semantic associations (Anyanwu and Sheth, 2003) between entities in ontologies has been a major focus for the semantic web. These associations capture the complex relationships between entities that might be involve several other entities and can't be easily captured by human users in the midst of a large dataset. If a

S. Tartir (✉)

Faculty of Information Technology, Philadelphia University, Amman 19392, Jordan
e-mail: startir@philadelphia.edu.jo

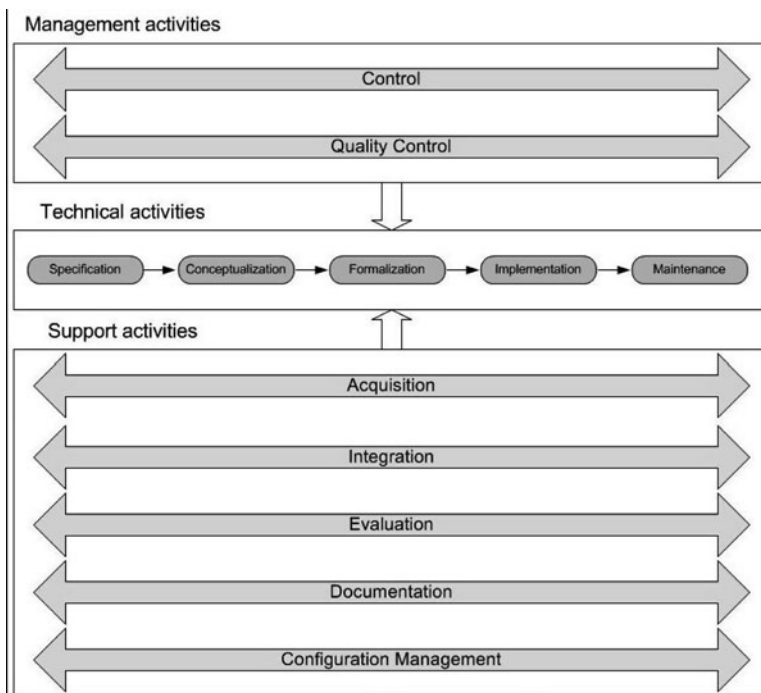


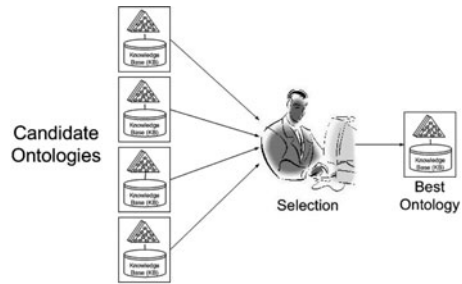
Fig. 5.1 Ontology life-cycle in meth ontology

user is interested in this type of search, she will also be interested to know about the presence of clusters of instances in the ontology, or the lack of a diverse set of relationships that might be of importance to her, because this knowledge will directly affects results return by this type of search.

In addition to the need during the process of building an ontology, evaluation and validation of ontologies are also useful for end users in domains where several ontologies with similar areas of interest are common. For example, many ontologies have been created for bioinformatics, and a researcher building an application that utilizes ontologies about genes might use an ontology search engine [e.g. Swoogle (Finin et al., 2005)] or an ontology library (e.g. Protégé Ontologies Library [Protégé]) find an ontology that best fit his research (e.g. MGED [MGED], GO [GO], OBO [OBO]) but will often find several ontologies that cover genes and it will be difficult for the user to simply glance through the resulting ontologies to find the most suitable ontology. In this and similar domains, a tool that would provide an insight into the ontology and describe its features in a way that will allow such a researcher to make a well-informed decision on the ontology that best fits her needs is needed (Fig. 5.2).

The OntoQA (Tartir et al., 2005) technique we present in Section 5.4 for ontology analysis and evaluation was developed after facing some of the issues presented above in the continuing process of building SWETO (the Semantic Web Technology

Fig. 5.2 Selecting the “best” ontology



Evaluation Ontology; Aleman-Meza et al., 2004). SWETO is a large-scale general purpose ontology that was populated with hundreds of thousands of instances that were mostly automatically extracted from different resources. This population process introduced a few problems. For example, SWETO includes some knowledge about geographical entities, like countries, states, and cities, and it was noticed that due to the nature of the sources instances were extracted from, that most of relationships extract were instances of the “located_in” relationship, or that most of the instances were about authors and publications. These and similar problems will prevent, for example, the discovery of interesting and useful relationships that connect people by their presence at the same location at the same time. Such problems would lower the efficiency and usefulness of some of the Semantic Web techniques such as the Semantic Association search mentioned above.

The rest of this chapter is organized as follows: Section 5.2 illustrates the need of ontology evaluation and validation. Section 5.3 introduces the current approaches in ontology evaluation and validation. Section 5.4 describes the OntoQA technique for ontology quality evaluation. Finally, Section 5.5 draws some conclusions and future recommendations.

5.2 Current Approaches in Ontology Evaluation and Validation

The increasing interest in the Semantic Web in recent years resulted in creating a large number of ontologies, and in increasing the amount of research on techniques to evaluate ontology quality and validity. With this growth in the number of ontologies, there have been some attempts to study the different approaches and tools for ontology evaluation and validation (Hartmann et al., 2004). Below is a description of the major current approaches currently in use for the evaluation and validation of ontologies.

5.2.1 Evolution-Based

This approach tracks an important characteristic of ontologies, change over time. Ontologies change over time by nature. More knowledge is always added to

domains, and it needs to be properly added to ontologies that model these domains. This approach tracks changes in the same ontology across different versions to get an indication of the quality of the ontology, and to detect (and possibly recover) any invalid changes made to the ontology. Ontologies change over time (evolve) due to three causes as proposed in Noy and Klein, (2004):

1. Changes in the domain,
2. Changes in conceptualization,
3. Changes in the explicit specification.

Changes in the domain are the most common, and are caused by change or addition in knowledge in the domains the ontology is modeling. For example, the more information about the genetic structure of a certain species are discovered, they need to be added to the ontology that models it.

Changes in conceptualization can result from a changing view of the world and from a change in usage perspective. Different tasks may imply different views on the domain and consequently a different conceptualization. For example, the view of a university from a faculty perspective is much different than the view from a student perspective, and the adoption of a certain perspective may result in a change of the ontology.

Changes in the explicit specification occur when an ontology is translated from one knowledge representation language to another. The languages differ not only in their syntax, but also (and more importantly) in their semantics and expressivity. Therefore, preserving the semantics of an ontology during translation is a non-trivial task.

An example of this approach is the technique presented in Plessers and De Troyer (2005). In this technique, when a change is needed on the ontology, a request is first added to the change log using CDL (Change Definition Language). Then, the change is implemented in the ontology. The technique finally matches the actual change with the change request from the log, if they are the same, the change is considered valid and it is propagated.

The technique in Haase et al. (2005) detects the two types of inconsistencies in evolving ontologies (user-defined and language-based) and repairs inconsistencies in ontologies across the different versions of the ontology by eliminating the statements that cause inconsistency.

5.2.2 Logical (Rule-Based)

Logical and rule-based approaches to ontology validation and quality evaluation use rules which are built in the ontology languages and rules users provided to detect conflicts in ontologies. Examples of first type are when two objects in an OWL ontology are said to be different from each other (`owl:differentFrom`), the ontology can't say that they are the same thing (`owl:sameAs`), or when two classes are said to

be disjoint of each other (`owl:disjointWith`) and the ontology can not have statements that mention an instances as being a member to both classes. Users can also identify properties that are considered in conflict in the domain. For example, a user can define that property `motherOf` conflicts with property `marriedTo`.

Several applications have adopted this approach. In Arpinar et al. (2006), a rule-based technique to conflict detection in ontologies is introduced. In this approach users identify conflicting rules using RuleML Boley et al. (2001) and the application will then list any cases were these rules are violated.

Authors of Parsia et al. (2005) use a logic model they call Swoop to detect unsatisfiable concepts in OWL ontologies. The technique is intended to be used by ontology designers to evaluate the quality of their work and to indicate any possible problems.

5.2.3 *Metric-Based (Feature-Based)*

Metric-based techniques to evaluate ontologies offer a quantitative perspective of ontology quality. These techniques scan through the ontology to gather different types of statistics about the knowledge presented in the ontology, or ask the user to input some information that is not included in the ontology itself. These techniques might consider classes' locations in the ontology schema graph as an indication of the type of knowledge the ontology focuses on. Some techniques also consider the instances of populated ontology in the measurement of quality metrics. The distribution of instances on the classes of the schema might also give an indication on the quality of the ontology.

Several techniques have adopted this approach. The authors of Lozano-Tello and Gomez-Perez (2004) propose a hierarchical framework they call *OntoMetric* that consists of 160 characteristics spread across five dimensions to evaluation the quality and suitability of ontologies to users' system requirements. The dimensions defined are: content of the ontology, language, development methodology, building tools, and usage costs. Users of *OntoMetric* will have the major task of supplying the application with several values that will be used to measure the suitability of an ontology for the given system requirements.

In Supekar et al. (2004) the authors propose a model for evaluating ontology schemas. The model contains two sets of features: quantifiable and non-quantifiable. Their technique is based on crawling the web to search for ontologies and store them locally, and then use information provided by the user, like the domain and weights for their proposed metrics to return the most suitable ontology.

Alani et al. (2006) presents a technique called *AKTiveRank* that finds a set of related ontologies to a set of terms the user enters. It uses an aggregation of the values of the four measures *AKTiveRank* includes to evaluation ontology schemas to select one of the ontologies to be the most suitable. The measures they developed are: class match, density, semantic similarity, and betweenness.

Corcho et al. (2004) introduce the *ODEval* tool that can be used for the automatic detection of possible syntactical problems in ontologies, such as the existence of

cycles in the inheritance tree of the ontology classes, inconsistency, incompleteness, and redundancy of classes and instances.

Mostowfi and Fatouhi (2006) define eight features they use to measure the quality of ontologies. These features are used to define a set of transformations to improve the quality of ontologies. For example, the authors suggest if a class (Student) has a property (Salary) that does not always have values (because it only holds for student assistants), then the class needs to be split into two: Student and Student Assistant. Other transformations attempt to make changes in properties or data types to make the ontology more consistent.

Another example technique is oQual (Gangemi et al., 2006), which evaluates ontologies on three dimensions: Structural: which uses a set of 32 features to study the syntax and formal semantics of the ontology. Functional: which uses a set of five qualitative measures to study the relationship between the ontology and its intended meaning. And finally, Usability profiling: which focuses on the communication (annotation) context of the ontology.

OntoClean (Guarino and Welty, 2004) also follows a feature-based approach to ontology evaluation and validation. A user of this technique would assign a set of four features to each of class in the ontology (Rigidity, Identity, Unity, and Dependence) and then use these features to identify problematic areas that needs to be reexamined. Based on these four features, classes might move up or down the taxonomy, and new classes might be added or removed to correct problems discovered through the detection of violations of a set of rules built using the four features.

The OntoQA framework we introduced in the abstract is one of the metric based approaches as well. In OntoQA we define the quality of a populated ontology as a set of five schema quality features and nine knowledgebase (or instance-base) quality features. An overview of OntoQA is presented in the next section.

Table 5.1 below provides a summary of the techniques mentioned above. The table compares the techniques on whether they target developers or end-users, whether users have to provide information to the technique (which might affect the training needed to be able to use the technique), whether it targets the schema or both the schema and the knowledgebase (KB), and whether users have to provide the ontologies or the application would crawl the internet for candidates.

It can be seen that among the techniques studied, most of them:

- Only work with schemas: This might miss problems and ignore knowledge available in the KB of a populated ontology.
- Require the user to provide the ontology: This might be problematic for a novice end-user who is not aware of ontologies available for his domain.
- Target developers (rather than end-users): Although evaluation and validation are important during the development process, it is important to provide end-users with tools they can use to select an error-free ontology that best fits their applications.
- Are feature-based: this is possibly due to the fact that a combination of metrics can provide insights about an ontology from different perspectives leading to a better understanding of the nature of the ontology.

Table 5.1 Comparison of different ontology evaluation techniques

Technique	Approach	Users	Automatic/ manual	Schema/KB	Ontology
Plessers and De Troyer (2005)	Evolution	Developers	Manual	Schema	Entered
Haase et al. (2005)	Evolution	Developers	Manual	Schema	Entered
Arpinar et al. (2006)	Logical	Developers	Manual	Schema + KB	Entered
Swoop	Logical	Developers	Automatic	Schema	Entered
OntoMetric	Metric	Developers	Manual	Schema	Entered
Supekar et al. (2004)	Metric	D + E	Automatic	Schema	Crawled
AKTiveRank	Metric	D + E	Automatic	Schema	Crawled
Mostowfi and Fatouhi (2006)	Metric	Developers	Automatic	Schema	Entered
oQual	Metric	D + E	Manual	Schema	Entered
OntoClean	Metric	Developers	Manual	Schema	Entered
OntoQA	Metric	D + E	Automatic	Schema + KB	Entered

Several researchers have studied the current approaches for ontology evaluation and validation. For example, Gómez-Pérez and Suarez-Figueroa (2003) compared several DAML/OIL and RDF(S) ontology checkers, validators, parsers and platforms (e.g. OilEd, OntoEdit, etc) and showed how most of the current tools were unable to find errors in ontologies. The authors also compared the tools with respect to three major problematic aspects: inconsistency, incompleteness, and redundancy. They concluded that tools that detect these errors are important for ontologies to be used more often.

5.3 OntoQA: Metric-Based Ontology Quality Analysis

In this section we describe OntoQA, our ontology evaluation tool. As mentioned in the previous section, OntoQA is a feature-based method for the evaluating ontologies (Fig. 5.3). OntoQA’s main characteristic that distinguishes it from other

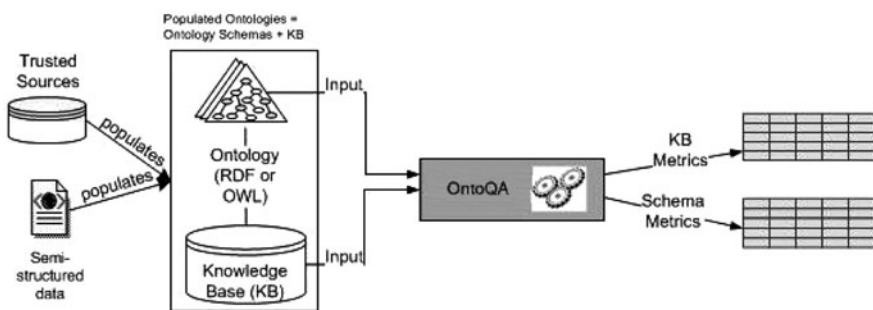


Fig. 5.3 OntoQA architecture

ontology quality tools is that it works on populated ontologies, thus enabling it from utilizing knowledge represented in the instances to gain a better measure of the quality of the ontology. OntoQA also uses much simpler techniques compared to others in that it doesn't require a lot of training as user involvement is minimal. In OntoQA, metrics (features) are divided into two groups: schema metrics that address the design of the ontology schema and instance metrics that address the way instances are organized within the ontology.

Metrics proposed in OntoQA describe certain aspects of the ontology rather than describing an ontology as merely "effective or ineffective" or "good or bad", because, in most cases, the way the ontology is built is largely dependent on the domain in which it is designed. For example, ontologies modeling human activities (e.g., travel or terrorism) will have distinctly different characteristics from those modeling the natural (or physical) world (e.g. genes or complex carbohydrates).

We divided the metrics into two related categories: schema metrics and knowledgebase (instance) metrics. The first category evaluates ontology design and its potential for rich knowledge representation. The second category evaluates the placement of instance data within the ontology and the effective utilization of the knowledge modeled in the schema. Below is a description of both categories of metrics.

5.3.1 Schema Metrics

Schema metrics address the design of the ontology. Although we cannot definitely know if the ontology design correctly models the domain knowledge, metrics in this category indicate the richness, width, depth, and inheritance of an ontology schema design. The most significant metrics in this category are described next.

5.3.1.1 Relationship Richness

This metric reflects the diversity of the types of relations in the ontology. An ontology that contains only inheritance relationships usually conveys less information than an ontology that contains a diverse set of relationships. The relationship richness is represented as the percentage of the (non-inheritance) relationships between classes compared to all of the possible connections that can include inheritance and non-inheritance relationships.

Definition 1: The relationship richness (RR) of a schema is defined as the ratio of the number of (non-inheritance) relationships (P), divided by the total number of relationships defined in the schema (the sum of the number of inheritance relationships (H) and non-inheritance relationships (P)).

$$RR = \frac{|P|}{|H| + |P|}$$

5.3.1.2 Inheritance Richness

Inheritance Richness (IR) measure describes the distribution of information across different levels of the ontology's inheritance tree or the fan-out of parent classes. This is a good indication of how well knowledge is grouped into different categories and subcategories in the ontology. This measure can distinguish a horizontal ontology (where classes have a large number of direct subclasses) from a vertical ontology (where classes have a small number of direct subclasses). An ontology with a low inheritance richness would be of a deep (or vertical) ontology, which indicates that the ontology covers a specific domain in a detailed manner, while an ontology with a high IR would be a shallow (or horizontal) ontology, which indicates that the ontology represents a wide range of general knowledge with a low level of detail.

Definition 2: The inheritance richness of the schema (IR) is defined as the average number of subclasses per class.

$$IR = \frac{|H|}{|C|}$$

5.3.1.3 Attribute Richness

The number of attributes (slots) that are defined for each class can indicate both the quality of ontology design and the amount of information pertaining to instance data. In general we assume that the more slots that are defined the more knowledge the ontology conveys.

Definition 3: The attribute richness (AR) is defined as the average number of attributes (slots) per class. It is computed as the number attributes for all classes (att) divided by the number of classes (C).

$$AR = \frac{|att|}{|C|}$$

5.3.2 Knowledgebase Metrics

The way data is placed within an ontology is also a very important measure of ontology quality because it can indicate the effectiveness of the ontology design and the amount of real-world knowledge represented by the ontology. Instance metrics include metrics that describe the KB (Knowledgebase) as a whole, and metrics that describe the way each schema class is being utilized in the KB.

5.3.2.1 Class Richness

This metric is related to how instances are distributed across classes. The number of classes that have instances in the KB is compared with the total number of classes, giving a general idea of how well the KB utilizes the knowledge modeled by the schema classes. Thus, if the KB has a very low Class Richness, then the KB does not have data that exemplifies all the class knowledge that exists in the schema. On the other hand, a KB that has a very high CR would indicate that the data in the KB represents most of the knowledge in the schema.

Definition 4: The class richness (CR) of a KB is defined as the percentage of the number of non-empty classes (classes with instances) (C') divided by the total number of classes defined in the ontology schema (C).

$$CR = \frac{|C'|}{|C|}$$

5.3.2.2 Class Connectivity

This metric is intended to give an indication of what classes are central in the ontology based on the instance relationship graph (where nodes represent instances and edges represent the relationships between them). This measure works in tandem with the importance metric mentioned next to create a better understanding of how focal some classes function. This measure can be used to understand the nature of the ontology by indicating which classes play a central role compared to other classes.

Definition 5: The connectivity of a class ($\text{Conn}(C_i)$) is defined as the total number of relationships instances of the class have with instances of other classes (NIREL).

$$\text{Conn}(C_i) = |\text{NIREL}(C_i)|$$

5.3.2.3 Class Importance

This metrics calculates the percentage of instances that belong to classes at the inheritance subtree rooted at the current class with respect to the total number of instances. This metric is important in that it will help in identifying which areas of the schema are in focus when the instances are added to the KB. Although this measure doesn't consider the domain characteristics, it can still be used to give an idea on what parts of the ontology are considered focal and what parts are on the edges.

Definition 6: The importance of a class ($\text{Imp}(C_i)$) is defined as the percentage of the number of instances that belong to the inheritance subtree rooted at C_i in the KB ($\text{inst}(C_i)$) compared to the total number of class instances in the KB (CI).

$$\text{Imp}(C_i) = \frac{|\text{Inst}(C_i)|}{|\text{KB}(CI)|}$$

5.3.2.4 Cohesion

In a semantic association discovery, relationships between instances are traced to discover how two instances are related. If the instances have disconnections among themselves, this may hinder such a search. This metric can be used to indicate the existence of such cases where the KB has more than one connected component (one being the ideal situation where all instances are connected to each other), indicating areas that need more instances in order to enable instances from one connect component to connect to instances in other connected components.

Definition 7: The cohesion (*Coh*) of a KB is defined as the number of connected components (*CC*) of the graph representing the KB.

□

5.3.2.5 Relationship Richness

This is an important metric reflecting how much of the relationships defined for the class in the schema are actually being used at the instances level. This is another good indication of the utilization of the knowledge modeled in the schema.

Definition 8: The relationship richness (*RR*) of a class C_i is defined as the percentage of the number of relationships that are being used by instances I_i that belong to C_i ($P(I_i, I_j)$) compared to the number of relationships that are defined for C_i at the schema level ($P(C_i, C_j)$).

In addition to these eight metrics (Tartir et al., 2005), includes other metrics that evaluate the ontology on other design aspects.

□

5.3.3 OntoQA Results

Figures 5.4 and 5.5 and Table 5.2 below show the OntoQA results when it is run on the three ontologies: SWETO (a general-purpose ontology with a focus on scientific publications), TAP (Guha and McCool 2003) (a general-purpose ontology) and GlycO (Sheth et al., 2004) (an ontology for the field of glycomics). It can be seen how different each one is by looking at the classes most instances in the ontology's KB fall into.

Figure 5.4 shows the most important classes in each of the ontologies. From the figure, it can be clearly seen that classes related to publications are the dominant classes in SWETO. While, with the exception of the Musician class, TAP gives consistent importance to most of its classes covering the different domains

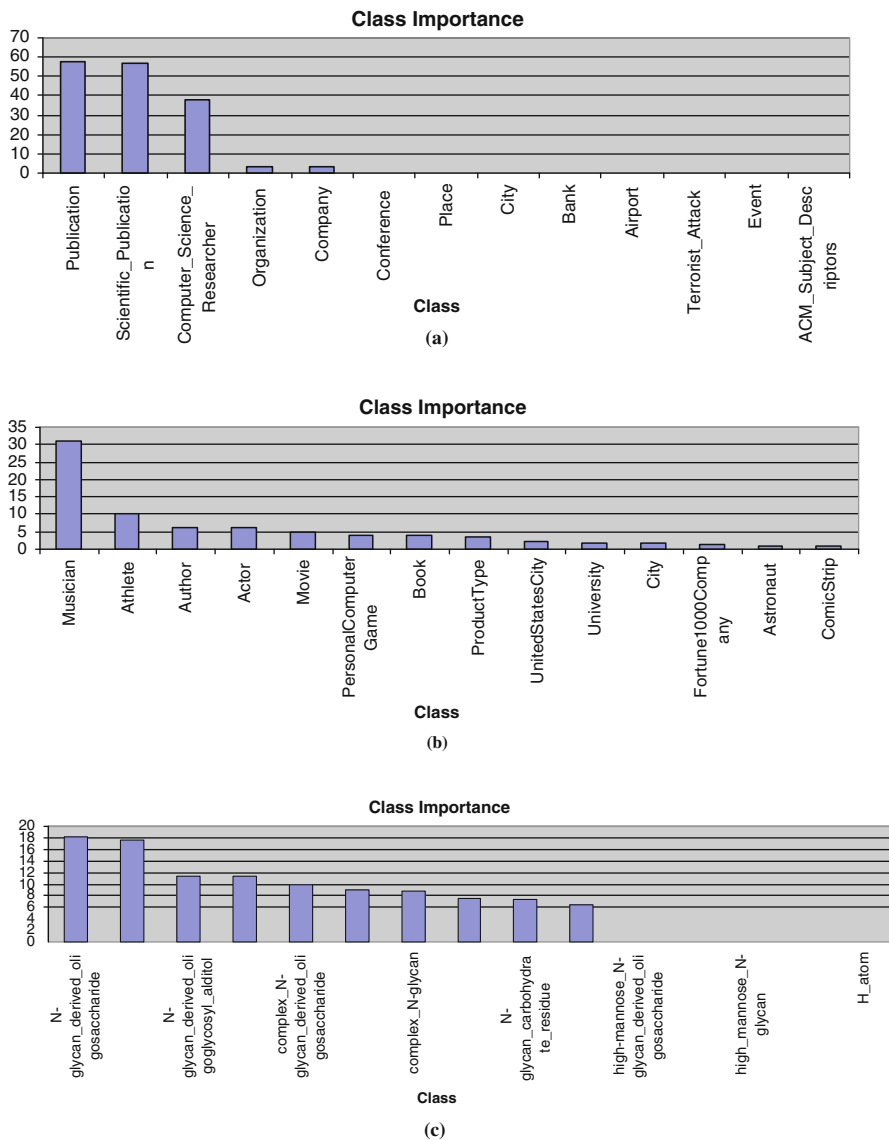


Fig. 5.4 Class importance in (a) SWETO (b) TAP and (c) GlycO

it includes. The nature of the GlycO ontology is reflected in the classes that are most important. The importance of the “N-glycan_residue” and the “alpha-D-mannopyranosyl_residue” and other classes show the narrow domain of GlycO is intended for, although the “glycan_moiety” class is the most important class covering about 90% of the instances in the KB.

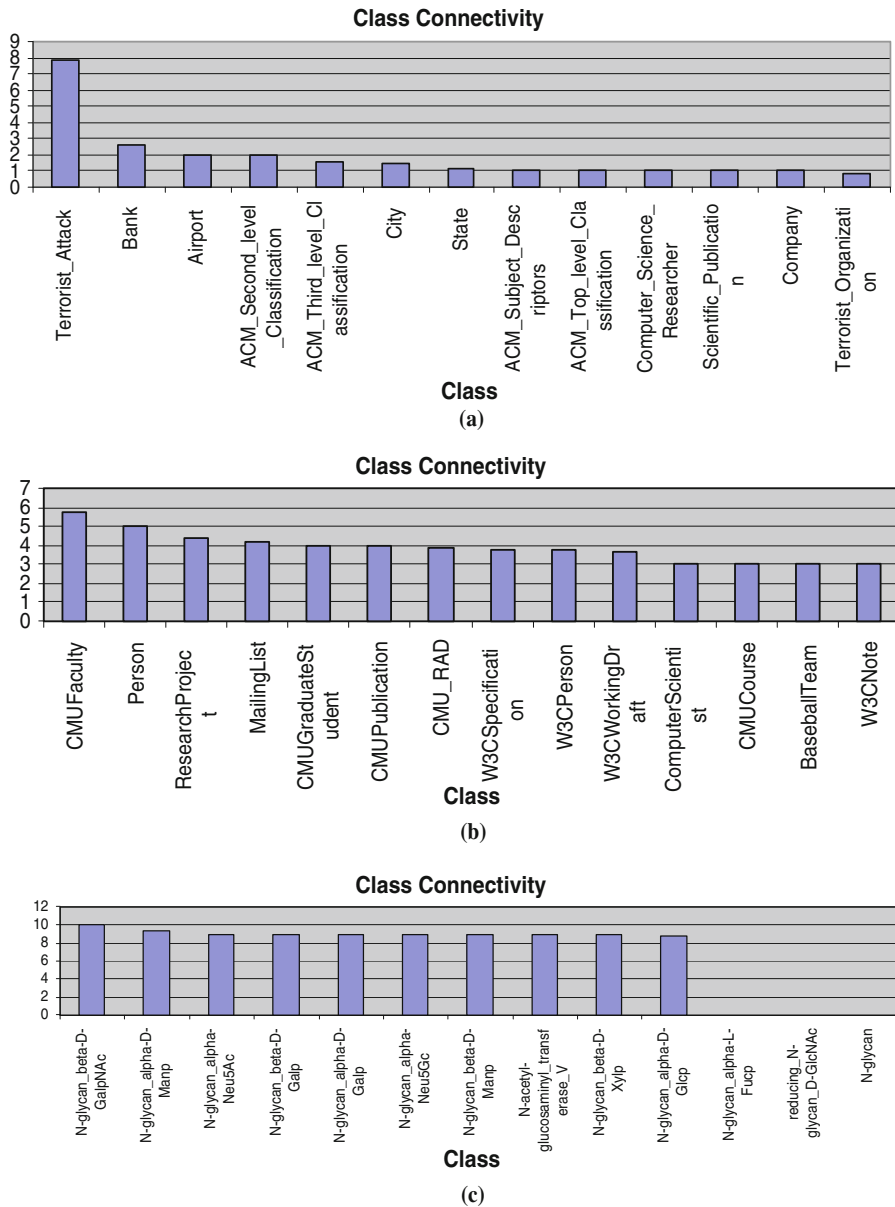


Fig. 5.5 Class connectivity in (a) SWETO (b) TAP and (c) GlycO

Figure 5.5 shows the most connected classes in the three ontologies. From the figure, it can be seen that SWETO also includes good information about domains other than publications, including the terrorism domain (Terrorist_Attack and Terrorist_Organization), the business domain (Bank and Company) and geographic

Table 5.2 Summary of SWETO, TAP, and GlycO

Ontology	Classes	Relations	Instances	Class richness
SWETO	44	101	813,217	59.1
TAP	6,959	25	85,637	0.24
GlycO	361	56	660	48.1

information (City and State). In a similar manner, TAP continues to show that it covers different domains, and its most connected classes cover the education domain (CMUCourse and CMUSCS_ResearchArea), the entertainment domain (TV and Movie), and other domains as well. GlycO's specific-purpose nature is evident from the Glycan related classes that are most connected.

Table 5.2 shows the differences between the three ontologies by the number classes, relationships, and instances, and by their class richness metric, which indicate that more of SWETO's classes are populated with instances compared to TAP or GlycO, which may indicate that the instance population process was carried out to cover resources that reflect the diversity of knowledge in the schema.

5.4 Conclusion

Ontologies form the cornerstone of the Semantic Web, and as the Semantic Web gains acceptance of the different scientific domains, more ontologies will be created to capture and share the knowledge in these domains. With this comes the need of being able to evaluate and validate these ontologies to ensure that they correctly represent the domain knowledge, and to be able to select the ontology among different ontologies that best fits a certain application. In this chapter we have summarized the current major trends in evaluating and validating ontologies and given examples techniques of each trends. We also presented our work in *OntoQA* and shown how it can be used to evaluate the ontology across different dimensions to give accurate metrics describing the ontology.

Still, more work is needed in ontology evaluation and validation to have techniques that can help the user by searching for ontologies instead of requiring the user to provide one, and have more techniques that target end-users in addition to developers.

References

- Alani, H., C. Brewster, and N. Shadbolt. 2006. Ranking ontologies with *aktiverank*. In *Proceedings of the 5th International Semantic Web Conference*, Athens, GA, 5–9 Nov 2006.
- Aleman-Meza, B., C. Halaschek, A. Sheth, I.P. Arpinar, and G. Sannapareddy. 2004. SWETO: Large-scale semantic web test-bed. In *Proceedings of the 16th Seke 2004: Workshop on Ontology in Action*, Banff, AB, 21–24 June 2004, 490–493.

- Anyanwu, K., and A. Sheth. 2003. ρ -Queries: Enabling querying for semantic associations on the semantic web. In Proceedings of the 12th International. WWW Conference, Hungary.
- Arpinar, I.B., K. Giriloganathan, and B. Aleman-Meza. 2006. Ontology quality by detection of conflicts in metadata. In Proceedings of the 4th International EON Workshop, Edinburgh, 22 May 2006. Edinburgh; International Conference Center.
- Boley, H., S. Tabet, and G. Wagner. 2001. Design rationale of ruleml: A markup language for semantic web rules. In Proceeding of the 1st Semantic Web Working Symposium. Palo Alto, CA: Stanford University.
- Corcho, O. et al. 2004. ODEval: A tool for evaluating RDF(S), DAML+OIL, and OWL concept taxonomies. In Proceedings of the 1st IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI 2004), Toulouse, France, 369–382.
- Cristani, M., and R.A. Cuel. 2005. Survey on ontology creation methodologies. *International Journal of Semantic Web and Information Systems (IJSWIS)* 1(2):49–69.
- Fernández, M., A. Gómez-Pérez, J. Pazos, and A. Pazos. 1999. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems Applications*. 4(1):37–45.
- Finin, T., et al. 2005. Swoogle: Searching for knowledge on the semantic web. In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 05), Pittsburg, PA.
- Gangemi, A., C. Catenacci, M. Ciaramita, and J. Lehmann. 2006. Modelling ontology evaluation and validation. In Proceedings of the 2006 European Semantic Web Conference. Berlin: Springer.
- The Gene Ontology. <http://www.geneontology.org>
- Gómez-Pérez, A., and M. Rojas-Amaya. 1999. Ontological reengineering for reuse. In Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management, Dagstuhl Castle, Germany.
- Gómez-Pérez, A., and M.C. Suarez-Figueroa. 2003. Results of taxonomic evaluation of RDF(S) and DAML+OIL ontologies using RDF(S) and DAML+OIL validation tools and ontology platforms import services. In Proceedings of the 2nd International Evaluation of Ontology-Based Tools Workshop, 20th Oct 2003. Sanibel Island, FL: Sundial Resort.
- Guarino, N. and C. Welty. 2004. An overview of ontoclean. *Handbook on ontologie*, eds. S. Staab, and R. Studer, 151–159. Berlin: Springer.
- Guha, R., and R. McCool. 2003. TAP: A semantic web test-bed. *Journal of Web Semantics* 1(1): 81–87.
- Haase, P., F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y.A. Sure. 2005. Framework for handling inconsistency in changing ontologies. In Proceedings of ISWC2005, Galway, Ireland.
- Hartmann, J., P. Spyns, A. Giboin, D. Maynard, R. Cuel, M. Carmen Suárez-Figueroa, and Y. Sure. 2004. Methods for ontology evaluation. *Knowledge Web Deliverable*, D1.2.3, v. 0.1.
- Lozano-Tello, A., and A. Gomez-Perez. 2004. ONTOMETRIC: A method to choose the appropriate ontology. *Journal of Database Management* 15:1–18.
- MGED. The MGED Ontology. http://mged.sourceforge.net/ontologies/MO_FAQ.htm
- Mostowfi, F., and F. Fotouhi. 2006. Improving quality of ontology: An ontology transformation approach. In Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06), Atlanta, GA.
- Noy, N.F., and M. Klein. 2004. Ontology evolution: Not the same as schema evolution. In *Knowledge and information systems*.
- Open Biomedical Ontologies. <http://obo.sourceforge.net>
- Parsia, B., E. Sirin, and A. Kalyanpur. 2005. Debugging OWL ontologies. In Proceedings of WWW 2005, 10–14 May 2005, Chiba, Japan.
- Paslaru, E., B. Simperl, C. Tempich, and Y. Sure. 2006. ONTOCOM: A cost estimation model for ontology engineering. In Proceedings of 5th International Semantic Web Conference (ISWC 2006), Athens, GA.
- Plessers, P., and O. De Troyer. 2005. Ontology change detection using a version log. In Proceedings of the 4th International Semantic Web Conference (ISWC-05).

- Protégé Ontologies Library. <http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary>.
- Sabou, M., V. Lopez, E. Motta, and V. Uren. 2005. Ontology selection: Ontology evaluation on the real semantic web. In Proceedings of the 15th International World Wide Web Conference 2005, Edinburgh.
- Sheth, A., et al. 2004. Semantic web technology in support of bioinformatics for glycan expression. W3C Workshop on Semantic Web for Life Sciences, 27–28 Oct 2004, Cambridge, MA.
- Supekar, K., C. Patel, and Y. Lee. 2004. Characterizing quality of knowledge on semantic web. In Proceedings of AAAI Florida AI Research Symposium (FLAIRS-2004), 17–19 May 2004, Miami Beach, FL.
- Tartir, S., I.B. Arpinar, M. Moore, A.P. Sheth, and B. Aleman-Meza. 2005. OntoQA: Metric-based ontology quality analysis. In IEEE ICDM 2005 Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, 27 Nov 2005, Houston, TX.

Chapter 6

Tools for Ontology Engineering and Management

Lambrini Seremeti and Achilles Kameas

6.1 Introduction

Nowadays, ontologies are developed for and used in such diverse fields as qualitative modeling, language engineering, database design, information retrieval and extraction, knowledge management and organization, search and retrieval of information, e-commerce and configuration. The engineering part of developing ontologies comprises a complex set of activities that are conducted during conceptualization, design, implementation and deployment of ontologies. Ontology engineering covers a whole range of topics and issues, such as philosophical and metaphysical issues and knowledge representation formalisms, methodology of ontology development, recent Web technologies such as XML, RDF, OWL and their derivatives, business process modeling, common sense knowledge, systematization of domain knowledge, Internet information retrieval, standardization, evaluation, ontology integration with agents and applications, etc. It also helps in defining the essential concepts of the world of interest, provides design rationale for a knowledge base, supports a more disciplined design of a knowledge base and enables the accumulation of knowledge about it. As a consequence, the use of specific software tools that enable ontology conceptualization, representation, construction and use becomes an important aspect of building ontologies.

In this chapter, we first discuss a classification of ontology development and management tools (Section 6.2) according to the tasks that involve an ontology, for which we also provide some representative tools as examples. In Section 6.3, we discuss issues to consider when selecting an appropriate tool. Finally, Section 6.4 concludes this chapter with future, trends on tool, for ontology construction and evolution.

A. Kameas (✉)

Distributed Ambient Information Systems Group, Hellenic Open University & Computer Technology Institute, Patras, Hellas Greece
e-mail: kameas@eap.gr

6.2 Classification of Ontology Tools

The rapid growth of documents, web pages and other types of textual content pose a great challenge to modern content management systems. Ontologies offer an efficient way to reduce information overload by encoding the structure of a specific domain thus offering easier access to the information. There are numerous ontology engineering and management tools in use today. Most of them have resulted from efforts of research groups and university labs, so they are currently free. So far, there were several efforts to develop a comprehensive classification of ontology tools. One widely adopted taxonomy has been proposed by the OntoWeb Consortium (OntoWeb Consortium, 2002) and includes the following large categories: Ontology development tools, Ontology merge and integration tools, Ontology evaluation tools, Ontology-based annotation tools, Ontology storage and querying tools, and Ontology learning tools. In this chapter, we shall adopt a shallow classification of ontology tools in two large categories: specialized ontology engineering tools and integrated ontology engineering environments (see Fig. 6.1).

6.2.1 Specialized Ontology Engineering Tools

These are the tools that support a restricted set of activities of the entire ontology life-cycle (design, deployment, maintenance, evolution). They can be divided in ontology engineering tools, ontologies combination tools, and ontology management tools.

6.2.1.1 Ontology Engineering Tools

Ontology engineering is a set of tasks related to the development of ontologies for a particular domain. It aims at making explicit the knowledge contained within software applications, and within enterprises and business procedures for this domain. An ontology engineering tool is any tool used for creating ontologies or similar semantic documents. From this perspective, these tools can be classified in ontology building tools and ontology learning tools. The former help engineers construct an ontology from scratch, whereas the latter can be used for the (semi)automatic construction of an ontology.

Ontology Building Tools

An ontology building process includes problem specification, domain knowledge acquisition and analysis, conceptual design and commitment to community ontologies, iterative construction and testing, publishing the ontology as terminology, and possibly populating a conforming knowledge base with ontology individuals. While the process may be strictly a manual exercise, there are tools available that can automate portions of it. Some examples of the so-called ontology editors are CODE (<http://www.cs.indiana.edu/~treicche/code.pdf>), DOE

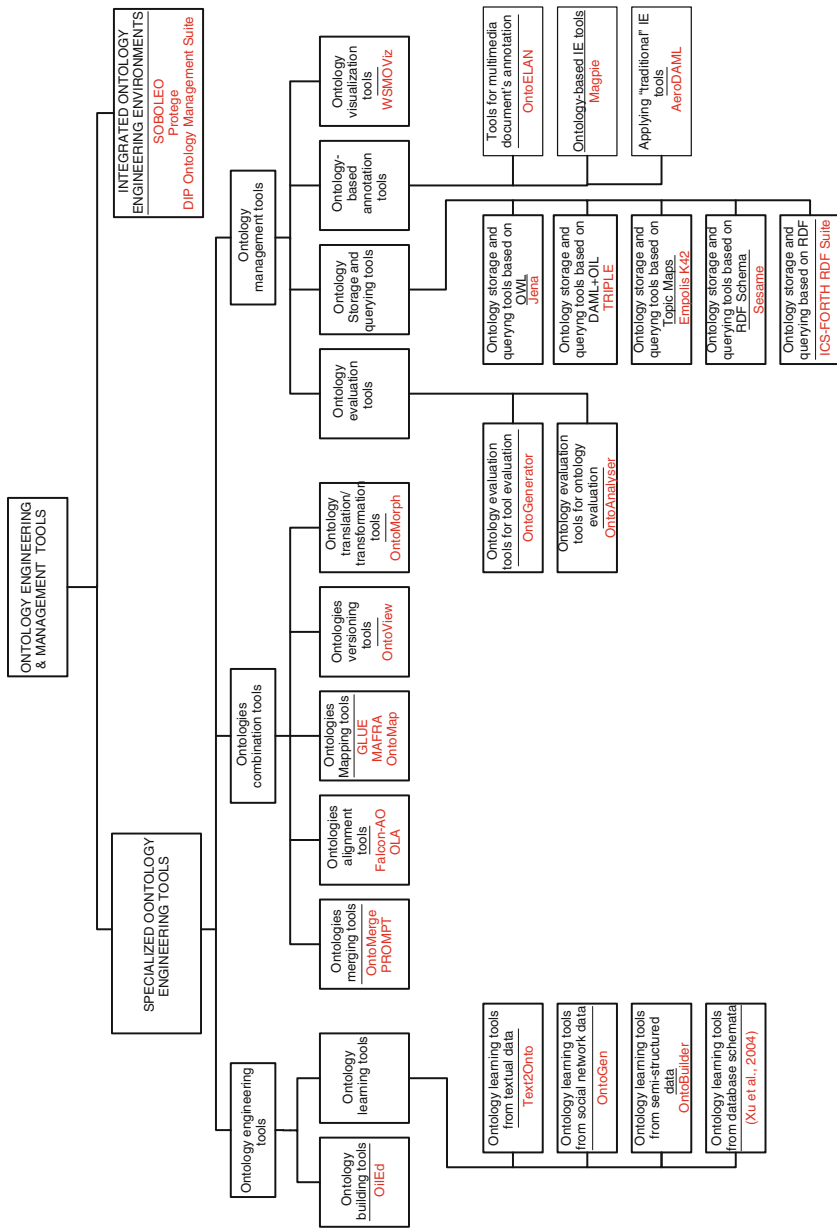


Fig 1. Taxonomy of ontology engineering and management tools and environments

(<http://homepages.cwi.nl/~troncy/DOE>), DUET (<http://codip.grci.com/Tools/Tools.html>), JOE (<http://www.engr.sc.edu/research/CIT/demos/java/joe>), ODE (<http://www.swi.psy.uva.nl/wondertools/html/ODE.html>), OilEd (<http://oiled.man.ac.uk>) and OntoSaurus (<http://www.isi.edu/isd/ontosaurus.html>).

OilEd (<http://www.redland.opensource.ac.uk/demo/>) is a graphical ontology editor developed by the University of Manchester that allows the user to build ontologies using DAML+OIL. The initial intention behind OilEd was to provide a simple editor that demonstrates the use of OIL language. Its current versions do not provide a full ontology development environment – it does not actively support the development of large-scale ontologies, the migration and integration of ontologies, versioning, argumentation and many other activities that are involved in ontology construction. Rather, it is the “NotePad” of ontology editors, offering enough functionality to allow users to build ontologies and to demonstrate how one can use the FaCT reasoner to check those ontologies for consistency.

Ontology Learning Tools

Ontology learning is a wide domain of research that consists of ontology enrichment, inconsistency resolution and ontology population. Ontology enrichment is the task of extending an existing ontology with additional concepts and relations and placing them in the correct position in the ontology. Inconsistency resolution is the task of resolving inconsistencies that appear in an ontology with the view to acquire a consistent (sub)ontology. Ontology population is the task of adding new instances of concepts into the ontology.

Acquiring domain knowledge for building ontologies requires much time and many resources. In this sense, one can define ontology learning as the set of methods and techniques used for building an ontology from scratch, or enriching, or adapting an existing ontology in a semi-automatic fashion using several sources. Several approaches exist for the partial automatization of the knowledge acquisition process. To carry out this automatization, natural language analysis and machine learning techniques can be used. Alexander Maedche and Steffen Staab (2001) distinguish different ontology learning approaches focused on the type of input used for learning. In this sense, they propose the following classification: ontology learning from text, from dictionary, from knowledge base, from semi-structured schemata and from relational schemata.

Depending on the different assumptions regarding the provided input data, ontology learning can be addressed via different tasks: learning just the ontology concepts, learning just the ontology relationships between the existing concepts, learning both the concepts and relations at the same time, populating an existing ontology/structure, dealing with dynamic data streams, simultaneous construction of ontologies giving different views on the same data, etc. More formally, the ontology learning tasks are defined in terms of mappings between ontology components, where some of the components are given and some are missing and one wants to induce the missing ones. We shall use a different classification of ontology learning tools, that is based on the input data provided.

Ontology Learning Tools from Textual Data

As human language is a primary means of knowledge transfer, ontology learning from relevant text collections has been among the most successful strategies towards developing and maintaining ontologies dynamically. More information on ontology learning from text can be found in Buitelaar et al. (2005). Some examples of systems for ontology learning from text are KEA (Jones and Paynter, 2002), OntoLearn (Velardi et al., 2005), Welkin (Alfonseca and Rodriguez, 2002), and Text2Onto (Ciniamo and Volker, 2005).

Text2Onto (Ciniamo and Volker, 2005) is a framework for ontology learning from textual resources. Three main features distinguish Text2Onto from its earlier framework TextToOnto, as well as other state-of-the-art ontology learning frameworks. First, by representing the learned knowledge at a meta-level in the form of instantiated modeling primitives within a so called Probabilistic Ontology Model (POM), it remains independent of a concrete target language, while being able to translate the instantiated primitives into any (reasonably expressive) knowledge representation formalism. Second, user interaction is a core aspect of Text2Onto and the fact that the system calculates a confidence for each learned object, allows designing sophisticated visualizations of the POM. Third, by incorporating strategies for data-driven change discovery, it avoids processing the whole corpus from scratch each time it changes, only selectively updating the POM according to the corpus changes instead. Besides increasing efficiency in this way, it also allows a user to trace the evolution of the ontology with respect to the changes in the underlying corpus.

Ontology Learning Tools from Social Network Data

Traditional Semantic Web deals with ontologies constructed mainly from text documents. Special ontology learning techniques deal almost exclusively with the problem of extracting and modeling the knowledge from text documents. The reason for this is that text is the most natural way of encoding information with the attached semantics. But text is not the only data modality which could be modeled using ontological structures. Ontological models can also be built from social network data. An example of these ontology learning tools is OntoGen (Fortuna et al., 2006).

OntoGen (Fortuna et al., 2006) is an ontology learning tool that can handle data represented as a set of feature vectors describing properties of ontological instances. Using several machine learning techniques (most prominent being k-means clustering, latent semantic indexing support vector machines, and uncertainty sampling active learning) OntoGen helps the user to construct an ontological structure directly from the data by providing suggestions and analyzing the user's decisions.

Ontology Learning Tools from Semi-structured Data

With the success of new standards for document publishing on the Web, there will be a proliferation of semi-structured data and formal descriptions of semi-structured data freely and widely available. HTML data, XML data, XML Document Type

Definitions (DTDs), XML-Schemata, and their likes add – more or less expressive – semantic information to documents. Ontologies can play a major role for allowing semantic access to these vast resources of semi-structured data. Though only few approaches do yet exist, extracting ontologies from these data and data descriptions may considerably leverage the application of ontologies and, thus, facilitate the access to these data. An example of this category is *OntoBuilder* (Modica et al., 2001).

The *OntoBuilder* (Modica et al., 2001) tool helps users in creating an ontology using as source semi-structured data coded in XML or in HTML. The modular architecture of the system consists of three main modules: the user interaction module, the observer module, and the ontology modeling module. The process followed to build an ontology has two phases: the training and the adaptation phase. In the training phase, an initial domain ontology is built using the data provided by the user. The adaptation phase aims in refining and generalizing the initial ontology. The user suggests browsing other web sites that contain relevant information for the domain. From each site, a candidate ontology is extracted and merged with the existing ontology. To perform this activity, a thesaurus can be used.

Ontology Learning Tools from Database Schemata

Ontologies play a key role in creating machine-processable Web content in order to promote Semantic Web. Extracting domain knowledge from database schemata can profitably support ontology development, as the Entity-Relationship (ER) model is an industrial standard for conceptually modeling databases.

In Xu et al. (2004) a formal approach is presented, as part of an automated tool for translating ER schemata into Web ontologies in the OWL Web Ontology Language. The tool can firstly read in an XML-codec ER schema produced with ER CASE tools such as PowerDesigner. Following the predefined knowledge-preserving mapping rules from ER schema to OWL DL (a sublanguage of OWL) ontology, it then automatically translates the schema into the ontology in both the abstract syntax and the RDF/XML syntax for OWL DL.

6.2.1.2 Ontologies Combination Tools

When one wants to reuse different ontologies together, those ontologies have to be *combined* in some way. This can be done by *integrating* the ontologies, which means that they are *merged* into a new ontology; alternatively, the ontologies can be *mapped*, that is, they can be kept separate. In both cases, the ontologies have to be *aligned*, which means that they have to be brought into mutual agreement. The alignment of concepts between ontologies is difficult, because it requires understanding of the meaning of concepts. Ontology alignment is concerned with the discovery of correspondences between ontologies. Ontology mapping is mostly concerned with the representation of these correspondences. Ontology merging is concerned with creating the union of ontologies, based on these correspondences. Aligning two ontologies, implies changes to at least one of them. Changes to an ontology will

result in a new *version* of an ontology. If the ontologies are not represented in the same language, a *translation* is often required. It is not possible to make a strict distinction between the tools used for each ontology combination problem, because each tool usually provides support for several types of combination.

Ontologies Merging Tools

Ontologies merging is the creation of an ontology from two or more source ontologies. The new ontology will unify and in general replace the original source ontologies. There are two distinct approaches in ontologies merging. In the first approach, the input of the merging process is a collection of ontologies and the outcome is a new, merged ontology, which captures the original ontologies. In the second approach, the original ontologies are not replaced, but rather a “view”, called *bridge ontology*, is created, which imports the original ontologies and specifies the correspondences using *bridge axioms*. *OntoMerge* (Dou et al., 2002), *PROMPT* (Noy and Musen, 2000), *FCA-Merge* (Stumme and Maedche, 2001), *HCONE-merge* (Kotis et al., 2006), and *IF-Map* (Kalfoglou and Schorlemmer, 2003) are tools that support the merging process.

OntoMerge (Dou et al., 2002) facilitates the creation of a “bridge” ontology, which imports the original ontologies and relates the concepts in these ontologies using a number of bridge axioms. It is an approach, in which the source ontologies are maintained after the merge operation. The output of the merge operation is not a completely merged ontology, but a bridge ontology which imports the source ontologies and has a number of Bridging Axioms, which are the translation rules used to connect the overlapping part of the source ontologies. It accepts a set of concepts or instance data based on one or more DAML ontologies, and a target ontology and produces the concepts or instance data translated to the target ontology.

PROMPT (Noy and Musen, 2000) is an algorithm and an interactive tool for the merging of two ontologies. It identifies a number of ontologies merging operations (merge classes, merge slots, merge bindings between a slot and a class, etc.) and a number of possible conflicts introduced by the application of these operations (name conflicts, dangling references, redundancy in the class hierarchy, and slot-value restrictions that violate class inheritance).

Ontologies Alignment Tools

Ontologies alignment is the process of discovering similarities between two source ontologies. It is generally described as the application of the so-called *Match* operator (Rahm and Bernstein, 2001). The input of the operator is a number of ontologies and the output is a specification of the correspondences between the ontologies. There are many different algorithms which implement the match operator. These algorithms can be generally classified along two dimensions. On the one hand, there is the distinction between schema-based and instance-based matching. A schema-based matcher takes different aspects of the concepts and relations in the ontologies and uses some similarity measure to determine correspondences (Noy and Musen, 2000). An instance-based matcher takes the instances which belong to

the concepts in the different ontologies and compares these to discover similarities between the concepts (Doan et al., 2004). On the other hand, there is the distinction between element-level and structure-level matching. An element-level matcher compares properties of the particular concept or relation, such as the name, and uses these to find similarities (Noy and Musen, 2000). A structure-level matcher compares the structure (e.g., the concept hierarchy) of the ontologies to find similarities (Giunchiglia and Shvaiko, 2004). These matchers can also be combined (Ehrig and Staab, 2004). Falcon-AO (Jian et al., 2005), OLA (Euzenat et al., 2004), and COMA++ (Aumueller et al., 2005) can be considered as ontologies alignment tools.

Falcon-AO (Jian et al., 2005) is an automatic tool for aligning ontologies. There are two matchers integrated in Falcon AO. One is a matcher based on linguistic matching for ontologies, called LMO. The other is a matcher based on graph-matching for ontologies, called GMO. In Falcon-AO, GMO takes the alignments generated by LMO as external input and outputs additional alignments. Reliable alignments are gained through LMO as well as GMO. Reliability is obtained by observing the linguistic comparability and structural comparability of the two ontologies being compared. Falcon-AO (version 0.6) copes not only with ontologies of moderate size, but also with very large-scale ontologies. It integrates three distinguishing elementary matchers, to manage different alignment applications, and the integration strategy is totally automatic.

OLA (Euzenat et al., 2004) is another alignment tool that follows the similarity-based paradigm. It is dedicated to the alignment of ontologies expressed in OWL, with an emphasis on its restricted dialect of OWL, called OWL-Lite. More than a simple tool for automated alignment construction, OLA is designed as an environment for manipulating alignments. Indeed, the system offers the following services: parsing and visualization of (pairs of) ontologies; automated computation of similarities between entities from different ontologies; automated extraction of alignments from a pair of ontologies; manual construction of alignments; initialization of automated alignment construction by an existing alignment; visualization of alignments; comparison of alignments.

Ontologies Mapping Tools

Ontologies mapping is an important step to achieve knowledge sharing and semantic integration in an environment in which knowledge and information have been represented with different underlying ontologies. The process of ontologies mapping is a (declarative) specification of the semantic overlap between two ontologies. Given two ontologies A and B, mapping one ontology with another, means that for each concept (node) in ontology A, we try to find a corresponding concept (node), which has same or similar semantics, in ontology B and vice versa (Castano et al., 2007). The three main phases for any mapping process are: mapping discovery; mapping representation; mapping execution. CROSI Mapping System (Kalfoglou and Hu, 2005), GLUE (Doan et al., 2004), MAFRA (Maedche et al., 2002), OntoMap[®] (Schnurr and Angele, 2005), and H-Match (Castano et al., 2006) are such tools.

GLUE (Doan et al., 2004) is a system which employs machine-learning technologies to semi-automatically create mappings between heterogeneous ontologies based on instance data, where an ontology is seen as a taxonomy of concepts. It focuses on finding 1-to-1 mappings between concepts in taxonomies, although the authors say that extending matching to relations and attributes and involving more complex mappings (such as 1-to-n and n-to-1 mappings) is the subject of ongoing research.

MAFRA (Mapping FRamework for distributed ontologies; Maedche et al., 2002) supports an interactive, incremental, and dynamic ontologies mapping process, the final purpose of which is to support ontology instance transformation. It adopts an open architecture in which concept mappings are realized through semantic bridges. A semantic bridge is a module that transforms source ontology instances into target ontology instances. The MAFRA toolkit supports a graphical user interface that provides domain experts with functionalities that are needed for the specification of semantic bridges. It has been implemented as a plug-in of KAON.

OntoMap[®] (Schnurr and Angele, 2005) is a plug-in for the ontology-management platform *OntoStudio*[®] that supports the creation and management of ontologies mappings. Mappings can be specified using a graphical representation, of the schema-view of the respective ontologies. It supports a number of elementary mapping patterns: concept to concept mappings, attribute to attribute mappings, relation to relation mappings, and attribute to concept mappings.

Ontologies Versioning Tools

As changes to ontologies are inevitable, it becomes very important to keep track of these changes, that is, the relation between successive revisions of one ontology and the relation between the ontology and its dependencies: instance data that conforms to the ontology; other ontologies that are built from, or import the ontology; applications that use the ontology. Therefore, a versioning tool is needed to handle revisions of ontologies and the impact on existing sources. In some sense, the versioning problem can also be regarded as a derivation of ontologies combination; it results from changes to individual ontologies. Although the problem is introduced by subsequent changes to a specific ontology, the most important problems are caused by the dependencies on that ontology. Ontologies versioning tools, such as SHOE (Heflin and Hendler, 2000), PROMPTDiff (Noy and Musen, 2002), *OntoView* (Klein and Fensel, 2001), *Ontology Versioning Tool* (De Leenheer et al., 2006), and *SemVersion* (Volkel and Groza, 2006) are about comparing ontology versions rather than about comparing independently developed ontologies.

OntoView (Klein and Fensel, 2001) is a system that helps ontology engineers to specify relations between ontology versions, in such a way, that interoperability between the versions of an ontology is improved. To provide a transparent interface to arbitrary versions of ontologies, *OntoView* keeps track of the conceptual relations and transformations between components of the ontology among different versions.

Such support is essential when ontologies are used on the Web and also useful for the collaborative development of ontologies.

Ontology Translation/Transformation Tools

Ontology transformation consists of transcribing an ontology from one form to another. This can include its expression in a different ontology language, or a reformulation in a restricted of a language (e.g., expressing automatically some non necessary OWL-Full ontology into OWL-DL), or with regard to a different vocabulary. Ontology transformation is useful for solving heterogeneity problems, when one wants to take advantage, in a particular context, of an ontology that has been developed in another context (i.e., using a different language). Ontology transformation is supported by a variety of tools. Some of them can be mere lexical or even be syntactic translators, but most will require the power of processing the ontology (i.e., inferring) in order to transform it. It is thus necessary to use the right tool: transformation systems are not version managers.

The term ontology translation is used in the literature to describe two different things. Under one understanding, ontology translation refers to the process of changing the formal representation of the ontology from one language to another (say from OWL to RDF or from Ontolingua to Prolog). This changes the syntactic form of the axioms, but not the vocabulary of the ontology. Works related to ontology translation under this understanding, leave the vocabulary of the ontology unaffected, dealing with the ontological axioms only. Under the second understanding, ontology translation refers to a translation of the vocabulary, in a manner similar to that of ontologies mapping. The difference between ontologies mapping and ontology translation is that the former specifies the function(s) that relate the two ontologies' vocabularies, while the latter applies this (these) function(s) to actually implement the mapping. *OntoMorph* (Chalupsky, 2000), WebODE translation system (Corcho, 2004), Transmorpher (Euzenat and Tardif, 2002), and RDFT (Omelayenko, 2002) are dedicated in ontology transformation.

OntoMorph (Chalupsky, 2000) is a system that supports the syntactic transformation of ontologies, using a language not very different from XSLT. It is however integrated with a knowledge representation system, which provides the ability to have semantically-grounded rules in the transformations. It uses syntactic rewriting via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and semantic rewriting. Syntactic rewriting is modulated via (partial) semantic models and logical inference.

6.2.1.3 Ontology Management Tools

Software tools are available to support the phases of ontology development. While ontology editors are useful during each step of the ontology building process, other types of ontology engineering tools are also needed along the way. Development projects often end up using numerous ontologies from external sources as well as existing and newly developed in-house ontologies. Ontologies from any source may

progress through a series of versions. As a consequence, the careful management of this collection of heterogeneous ontologies becomes necessary so as to keep track of them. Some tools also support ontology mapping and linking, comparison, reconciliation and validation, merging, and converting into other forms. Other tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology. Finally, other tools evaluate the existing ontologies, or use ontologies to annotate web documents, or support storing and querying ontologies. All these ontology management tools can be classified in four major categories that will be described in this section.

Ontology Evaluation Tools

Given the ever-increasing number of ontologies, there is an urgent need for evaluation, checking and validation tools. Ontology evaluation is the problem of assessing a given ontology from the point of view of a particular criterion of application, in order to determine if it would suit a particular purpose. It is a critical task, even more so when the ontology is the output of an automatic system, rather than the result of a conceptualization effort produced by a team of domain specialists and knowledge engineers. Ontology evaluation tools are responsible for checking ontology validation and consistency. These tools do not only validate the conformity to a standard of the syntactic representation in a special language, but also validate the semantic properties through the use of inference rules and other similar mechanisms.

Most evaluation approaches fall into four major categories: those based on comparing the released ontology to a predefined “golden standard”, which may be another hand-crafted ontology; those based on using the ontology in an integrated system and evaluating the performance of this system; those involving comparisons with a data source relative to the domain that is to be covered by the ontology and finally, those where the evaluation is performed purely by humans. Human assessment is performed by domain experts who try to assess how well the ontology meets a set of predefined criteria, requirements, standards, etc. Moreover, irrespective of the approach that is to be followed, an ontology can be evaluated at different layers. This is usually desirable, since an ontology is, in general, a complex structure and in most cases it is better to focus the evaluation on a particular aspect of the ontology.

Evaluation tools that evaluate the output ontology in a (semi)automatic way are needed, since the most widely used evaluation techniques are human-based and, consequently, may be biased. There also exist tools that evaluate the tools that develop ontologies.

Ontology Evaluation Tools for Tool Evaluation

Tools are evaluated for their technological properties, such as interoperability, performance, memory allocation, scalability, interface. An example of this category is *OntoGenerator* (Handschuh et al., 2001).

OntoGenerator (Handschuh et al., 2001) is an *OntoEdit* plug-in, focused on evaluating ontology tools’ performance and scalability.

Ontology Evaluation Tools for Ontology Evaluation

Ontologies are evaluated for their “language conformity” (i.e., to the syntax of a representation language) and for their “consistency” (i.e., to what extent they ensure consistency of specifications with respect to their semantics). *OntoAnalyser* (Handschuh et al., 2001), *OntoClean* (Guarino and Welty, 2002), *AEON* (Volker et al., 2005), *ONE-T* (Bouillon et al., 2002), and *CleanONTO* (Sleeman and Reul, 2006) are focused on ontology evaluation.

OntoAnalyser (Handschuh et al., 2001) is realized as a plug-in of *OntoEdit*. It focuses on evaluating ontology properties, particularly language conformity and consistency.

Ontology Storage and Querying Tools

Ontologies offer more services than simply representing knowledge and information. Human information consumers and web agents use and query ontologies and the resources committed to them. Since ontologies do not use traditional techniques and languages to represent information, the necessity to support adaptive storage and querying arises. However, the context of storing and querying knowledge has changed due to the wide acceptance and use of the Web as a platform for communicating knowledge. New languages for querying (meta)data based on web standards (e.g., XML, RDF, Topic Maps) have emerged, to enable the acquisition of knowledge from dispersed information sources, while the traditional database storage techniques have been adapted to deal with the peculiarities of the (semi)structured data on the web. So, these tools must provide full storage and query support to web-based ontology or metadata standards, such as RDF, RDFS, Topic Maps, DAML+OIL, and OWL. The tools presented here such as *ICS-FORTH RDFSuite* (Alexaki et al., 2001), *Sesame* (Broekstra et al., 2002), *Redland* (<http://www.redland.opensource.ac.uk/demo/>), *Jena* (McBride, 2001), *RACER* (<http://www.sts.tu-harburg.de/~r.f.moeller/racer>) are indicative of this tendency. We can classify these tools, according to the language that they are based on:

Ontology Storage and Querying Tools Based on OWL

Jena (McBride, 2001) is a Java framework that provides a programming environment for RDF, RDFS and OWL, including a rule-based inference engine. The *Jena2* release is more interesting for the ontology engineering process, as it provides an API that supports programmers who are working with ontology data based on RDF. This means that *Jena2* offers features for handling most of ontology languages, namely XML, OWL, DAML+OIL, RDFS. It also provides generic RDF Resource and Property classes to model more directly the class and property expressions found in ontologies using the above languages, and the relationships between these classes and properties. Due to its storage abstraction, *Jena* enables new storage subsystems to be integrated. In fact, *Jena* provides statement-centric methods

for manipulating an RDF model as a set of RDF triples, resource-centric methods for manipulating an RDF model as a set of resources with properties, as well as built-in support for RDF containers.

Ontology Storage and Querying Tools Based on DAML+OIL

TRIPLE (Sintek and Decker, 2001) is an inference and querying engine. It constitutes the implementation of the TRIPLE query language. It also contains a standalone DAML+OIL implementation with the following features: it parses DAML+OIL ontologies with Jena, provides output in various syntaxes, supports an external DL classifier that can be automatically invoked and its output can be returned in various formats.

Ontology Storage and Querying Tools Based on Topic Maps

Empolis K42 Knowledge Server constitutes a collaborative, web-based integrated authoring environment for capturing, expressing and delivering knowledge, which is able to import, export and merge Topic Maps (Magkanaraki et al., 2002).

Ontology Storage and Querying Tools Based on RDF Schema

Sesame (Broekstra et al., 2002) is an RDF Schema-based Repository and querying facility. It is a system consisting of a repository, a query engine and an administration module for adding and deleting RDF data and Schema information.

Ontology Storage and Querying Tools Based on RDF

ICS-FORTH RDFSuite (Alexaki et al., 2001) is a suite of tools for RDF metadata management, supporting RDF metadata processing for large-scale Web-based applications. It consists of tools for parsing, validating, storing and querying RDF descriptions.

Ontology-Based Annotation Tools

Ontology-based annotation refers to the process of creating metadata using ontologies as their vocabularies. Metadata is usually defined as “data about data”, which aims at expressing the “semantics” of information. It is used to describe documents and applications, in order to improve information seeking and retrieval and its understanding and use. Metadata can be expressed in a wide variety of vocabularies and languages, and can be created and maintained with a variety of tools. Ontology-based annotation tools are primarily designed to allow inserting and maintaining ontology based markups in Web pages. Most of these tools such as OntoMat Annotizer (Handschuh et al., 2001), SHOE Knowledge Annotator (Heflin and Hendler, 2000), GATE (Cunningham et al., 2002), Melita (Ciravegna et al., 2002), AeroDAML (Kogut and Holmes, 2001), Amilcare (Ciravegna and Wilks, 2003), MnM (Vargas-Vera et al., 2002), S-Cream (et al., 2002), Magpie (Domingue

et al., 2004), PANKOW system (Ciniamo et al., 2004), SemTag system (Dill et al., 2003), Photocopain (Tuffield et al., 2006), KIM system (Kiryakov et al., 2005), AKTive Media system (Chakravarthy et al., 2006), OntoELAN (Chebotko et al., 2004), and NOMOS (Niekrasz and Gruenstein, 2006) have appeared recently with the emergence of the Semantic Web. They use Information Extraction (IE) and Machine Learning (ML) techniques to propose semi-automatic annotations for Web documents.

Tools for Multimedia Documents' Annotation

Many language data are collected as audio and video recordings, which imposes a challenge to document indexing and retrieval. Annotation of multimedia data provides an opportunity for making the semantics of these data explicit and facilitates the searching for multimedia documents. So, a class of ontology-based annotation tools such as OntoELAN (Chebotko et al., 2004), AKTive Media system (Chakravarthy et al., 2006), Photocopain (Tuffield et al., 2006) are used as the ontology-based annotation tools for multimedia documents.

OntoELAN (Chebotko et al., 2004) is the first audio/video ontology-based annotation. It is an ontology-based linguistic annotator that features: support for loading and displaying ontologies specified in OWL; display a speech and/or video signals, together with their annotations; time linking of annotations to media streams; creation of a language profile, which allows a user to choose a subset of terms from an ontology and conveniently rename them if needed; creation of ontological tiers, which can be annotated with profile terms and, therefore, corresponding ontological terms; and saving annotations in XML format as multimedia ontology class instances and, linked with them, class instances of other ontologies used in ontological tiers.

Ontology-Based IE Tools

These tools perform semantic annotation with respect to an ontology without manual intervention (Bontcheva et al., 2006). Some examples of this category are PANKOW system (Ciniamo et al., 2004), KIM system (Kiryakov et al., 2005) and Magpie (Domingue et al., 2004).

Magpie (Domingue et al., 2004) is a suite of tools which supports the interpretation of web-pages and “collaborative sense-making”. It annotates web-pages with metadata in a fully automatic fashion without manual intervention, by matching the text against instances in the ontology. It automatically creates a semantic layer for web pages using a user-selected ontology. Semantic layers are annotations of a web page, with a set of applicable semantic services attached to the annotated items. It can automatically track concepts found during a browsing session using a semantic log. The log allows trigger services to be activated when a specific pattern of concepts has been found. The same log can be used as a conceptual representation of the user’s browsing history. Since all Magpie abilities are underpinned by ontological reasoning, this enables the users to use the history semantically rather than as a purely linear and temporal record of their activities.

Applying “Traditional” IE Tools

These tools do not incorporate ontologies into the system, but either use ontologies as a bridge between the IE system and the final annotation, as with AeroDAML (Kogut and Holmes, 2001), or rely on the user to provide the relevant information through manual annotation, as with Amilcare (Ciravegna and Wilks, 2003).

AeroDAML (Kogut and Holmes, 2001) is an annotation tool which applies IE techniques to automatically generate DAML annotations from web pages. It links most proper nouns and common types of relations with classes and properties in a DAML ontology. It consists of the AeroText IE system, together with components for DAML generation. A default ontology, which directly correlates to the linguistic knowledge base used by the extraction process, is used to translate the extraction results into a corresponding RDF model that uses the DAML+OIL syntax. This RDF model is then serialized to produce the final DAML annotation.

Ontology Visualization Tools

Ontologies are rich vocabularies, which implicitly contain more information than can be explicitly found in their text representation. Implicit information, such as the underlying structure of a data model, or which instances are most closely connected, is all contained in a graph. This information, though, is difficult, if not impossible, to extract from a text-based reading of the data. Tools that support the ontology visualization process are OntoViz (<http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz>), and WSMOViz (Kerrigan, 2006).

WSMOViz (Kerrigan, 2006) is an integrated ontology engineering and visualization tool for WSMO (Feier and Domingue, 2005). This tool does not only allow the user to view WSMO ontologies in a very clear way, but also to edit the ontology in the visual mode.

6.2.2 Integrated Ontology Engineering Environments

Ontology engineering, in addition to ontology construction, also supports mapping, management, maintenance and evolution of ontologies. There exist integrated collections of specialized tools that can support (fully or partially) more than one activities/processes of the ontology engineering life-cycle; these are called integrated ontology engineering environments. The need for these tools arises from the fact that the life cycle of ontology engineering is highly affected if the ontology is to be reused for building another ontology and vice-versa: different activities during the development of a specific ontology will be carried out if it is based on other ontologies that have already been built, or are under construction. This leads to interdependencies between the life cycles of ontologies. This interrelation between life cycles of several ontologies means that integration has to be approached globally (Fernandez-Lopez et al., 2000).

The integrated ontology engineering environments can represent all or some of the processes in the different phases of the ontology life cycle, such as the ontology generation (building, learning), ontology integration (merging,

mapping, versioning, translating), and ontology management (evaluating, annotating, storing, querying, visualizing). Integrated ontology engineering environments such as KAON (Bozsak et al., 2002), Protégé (Noy et al., 2000), WebODE (Aprirez et al., 2001), OntoEdit (Sure et al., 2002), SWOOP (Kalyanpur et al., 2005), HCONE (Kotis and Vouros, 2003), SOBOLEO (Zacharias and Braun, 2007), ORIENT (<http://apex.sjtu.edu.cn/projects/orient/>), NeOn Toolkit (<http://www.neon-toolkit.org>), TopBraid Composer (<http://www.topquadrant.com/topbraidvomposer.html>), OBO-Edit (<http://oboedit.org>), and DIP Ontology Management Suite (SALERO Consortium, 2006) can be classified in being web-based or not, in supporting collaborative ontology engineering or not, etc.

SOBOLEO (Zacharias and Braun, 2007) is the acronym for SOcial BOokmarking and Lightweight Engineering of Ontologies. The system's goal is to support people working in some domain in the collaborative development of a shared vocabulary and a shared index of relevant web resources. With SOBOLEO it is possible to create, extend and maintain taxonomies according to the SKOS Core Vocabulary in a simple way. It also supports the collection and sharing of relevant web resources (bookmarks). These bookmarks can be annotated with concepts from the SKOS taxonomy or arbitrary tags for better retrieval. One instance or installation of SOBOLEO is meant to be used by a community with interest in building a shared vocabulary and a web index. Within this one instance, users create and maintain collaboratively, one taxonomy and one shared index of web resources.

Protégé (Noy et al., 2000) was developed by Stanford Medical Informatics at the Stanford University School of Medicine. It was developed originally for use in the field of clinical medicine and the biomedical sciences, but now it is being used in many areas where the concepts can be modeled as a class hierarchy. It is a Java based open-source tool for editing and managing ontologies. It is the most widely used domain-independent, freely available, platform-independent technology for developing and managing terminologies, ontologies, and knowledge bases in a broad range of application domains. With Protégé it is easy to create classes and hierarchies, to declare properties for classes, create instances and introduce values, all these under an environment consisting in menus, buttons, dialog boxes and easy to use graphic representations. Its core functionality can be extended in many ways by creating plug-ins. There are over 90 Protégé plug-ins providing advanced capabilities such as reasoning and inference support and visualization of large ontologies. For example, PROMPTDiff (Noy and Musen, 2002) automatically calculates structural differences between ontology versions, identifying which concepts have changed between versions. It identifies both simple and complex changes and it presents the comparison results to the user in an intuitive way. Users then can accept or reject the changes between concepts and instances. Jambalaya is another Protégé plug-in that provides an extensible, flexible, and scalable visualization environment for exploring, navigating, and understanding ontologies. ONTO-H (Benjamins et al., 2004) is a tab plug-in for the Protégé ontology editor that allows the creation of annotations of RTF documents. OntoLT (Buitelaar et al., 2004) implements the definition of mapping rules, with which, classes and properties can be extracted automatically from linguistically annotated text collections. Through

the use of such rules, linguistic knowledge (context words, morphological and syntactic structure, etc.) remains associated with the constructed ontology and may be used subsequently in its application and maintenance, e.g., in knowledge markup, ontology mapping, and ontology evolution. OWLViz is also a Protégé plug-in which can be used to visualize ontologies built using the Protégé OWL plug-in. The OWL plug-in enables the creation and maintenance of ontologies described in the OWL syntax. A description logic inference engine, called RACER, is frequently used together with the Protégé OWL environment, as it provides reasoning services such as consistency checking and automated classification of concepts. Protégé can also support ontology merging with the PROMPT plug-in. The Protégé platform supports Web-based ontology viewing and collaboration and it provides different back-end storage mechanisms.

DIP Ontology Management Suite (<http://kmi.open.ac.uk/projects/dip/index.php#publications>) is an integrated set of tools for the effective management of ontologies, designed especially for handling and using ontologies as the underlying data model for Semantic Web Services. The whole suite consists of six major components that are developed as Eclipse plug-ins: Browsing and Editing (standard browsing and editing functionality); Mapping and Merging (allow to map and/or merge multiple ontologies); Versioning (controls the history of an ontology); Reporting (a graphical user interface for creating different types of diagram reports); Repository (for persistent storage and retrieval of ontology relevant data); Representation and Data Framework (middle layer and central API that provides transparent access to the suite's components).

6.3 Selecting the Appropriate Ontology Engineering And Management Tool

The continuous development of ontology editors and other tools for managing ontologies is an indication of the growing need for effective and universal knowledge representation in domains like the Semantic Web, Ubiquitous Computing, etc. These tools implement different knowledge models with different underlying knowledge representation paradigms, manage large upper level and general ontologies, and range from standalone to web-based and ubiquitous computing applications. Thus evaluation and comparison of these tools is important to help users determine which tool is best suited for their task.

In the last few years many studies evaluating ontology engineering tools have been published. Some authors have proposed general frameworks for the evaluation of ontology tools, i.e. the work presented by Duineveld and colleagues (Duineveld et al., 1999), the deliverable 1.3 of the OntoWeb project (2002), the conclusions attained in the First International Workshop on Evaluation of Ontology-based Tools (Angele and Sure, 2002), and Lambrix and colleagues (Lambrix et al., 2003). Others have presented more focused evaluations using specific criteria: Stojanovic and Motik (Stojanovic and Motik, 2002) analyzed the ontology evolution requirements fulfilled by the tools; Sofia Pinto and colleagues (Sofia Pinto et al., 2002)

evaluated the support provided by the tools in ontology reuse process; In the Second International Workshop on Evaluation of Ontology-based Tools (Corcho et al., 2003) the interoperability of the tools was evaluated; and Gomez-Perez and Suarez-Figueroa (Gomez-Perez and Suarez-Figueroa, 2004) evaluated the ability of the tools to detect taxonomic anomalies.

From all these studies, it's evident that, on one hand, there is no "one fits all" generic framework that can be used for comparing ontology engineering tools: different categories of tools require very different comparison frameworks. For example, ontology engineering tools (ontology building tools and ontology learning tools) can easily be compared since they all support similar tasks such as definition of concepts, instances, and relations in a domain. Ontology combination tools (ontology merging, alignment, mapping, versioning, translation tools) however are so different from one another that direct comparison may not be possible. They differ in the type of input they require (e.g., instance data or no instance data), the type of output they produce (e.g., one merged ontology, pairs of related terms, articulation rules), modes of interaction and so on. This diversity makes comparing the performance of ontology combination tools to one another largely meaningless. On the other hand, we can summarize certain criteria as the basis for the comparative evaluation of integrated ontology engineering environments, such as (http://www.ontoweb.org/download/deliverables/D21_Final-final.pdf):

- Extensibility: measures how adaptable these environments may be to future technological advances. It is crucial for preserving a full development evolution of integrated ontology engineering environments.
- Maturity: measures how integrated ontology engineering environments may handle development problems, and even reduce the number and intensity of the future problematic situations. This criterion comprises the ability to deal constructively with real environments, the capacity to adapt to change, the capacity to relate and combine with other integrated environments, and so on.
- Portability: the ability to adapt any integrated ontology engineering environment, technique or method within a new environment without redeveloping it.
- Interoperability: the ability of systems to operate in conjunction with each other, encompassing communication protocols, hardware, software applications, and data compatibility layers.
- Ease-of-use: covers ease-of-learning, intuitiveness, efficiency and functionality. Simultaneously measures how long it takes for one to learn to use a certain product, how intuitive the product is, and how logical it is to use, create or modify a program.

To be able to decide what ontology engineering tools are needed for fulfilling actual and future requirements of an application, one needs to objectively evaluate existing tools. For several reasons, this is a difficult task. Firstly, for an evaluation to be unbiased, it must be designed and carried out by someone other than tool developers themselves. Otherwise, the evaluation setup and comparison parameters are inevitably skewed (often subconsciously). Secondly, it is often hard to come up

with benchmark tasks and gold standards, because no two tools have been designed for the same purpose; thus, any attempt to evaluate a tool when performing restricted benchmark tasks often puts it in uses for which it was not designed. Thirdly, many of the criteria are, by their very nature, subjective. For example, when evaluating the quality of an ontology, we often don't have a single correct answer for how certain concepts should be represented or when evaluating the quality of ontology alignment, we often cannot agree on the precise relationships between concepts in source ontologies (<http://co4.inriaples.fr/align/contest>).

A systematic approach for comparing and selecting ontology engineering tools could be based on the taxonomy proposed in Fig. 6.1. Of course, this taxonomy should be enhanced with descriptions of specific properties of each class/tool describing various features and attributes of the class/tool, such as its architecture, the methodology that the tool follows, etc., as well as the allowed values for these properties. Then, available tools can be added as instances of specific classes (i.e., Protégé is an instance of the Integrated Ontology Engineering Environments class). In this classification, we have allocated each tool to one and only one class, but multiple inheritance should not be excluded, especially if we provide more abstract property descriptions.

This classification could be turned into an ontology of ontology engineering services, if we decompose tools into the functions each of them supports and then analyze the set of functions to form function classes. This ontology could then be questioned about what is the best ontology engineering tool for a specific application, in terms of the functions it supports, or which tools should be used to support the entire lifecycle of ontology engineering.

6.4 Conclusion

The next generation of semantic applications will be characterized by a large number of networked ontologies, some of them constantly evolving, most of them being locally, but not globally, consistent. In such scenarios, it is more or less infeasible to adopt current ontology management models, where the expectation is to have a single, globally consistent ontology, which serves the application needs of developers and possibly integrates a number of pre-existing ontologies. What is needed is a clear analysis of the complex relationships between ontologies in such networks, resulting in a formal model of networked ontologies that supports their evolution and provides the basis for guaranteeing their (partial) consistency in case one of the networked ontologies is changing. Open issues that are involved are among others ensuring consistency, evolution of ontologies and metadata, and reasoning. Developing tools that are able to meet these challenges is an essential requirement towards devising an ontology and metadata infrastructure that is powerful enough to support the realization of applications that are characterized by an open, decentralized, and ever changing environment.

Since ontologies encode a view of a given domain that is common to a set of individuals or groups in certain settings for specific purposes, mechanisms to

tailor ontologies to the need of a particular user in his working context are required. The efficient dealing with a user's context posts several research challenges, such as formal representation of context, context based reasoning and context mapping. A promising application area of contextual information is user profiling and personalization. Furthermore, with the use of mobile devices and current research on ubiquitous computing, the topic of context awareness is a major issue for future IT applications. Intelligent solutions are needed to exploit context information and rapidly changing environments and unsteady information sources. Advanced tools for assigning context to a situation have to be developed, which pave the way to introduce ontology-based mechanisms into context-aware applications.

An increasing number of application scenarios depend on the integration of information from various sources that comes in different formats and is characterized by different formalization levels. For example, in many large engineering companies, information can be typically found in text documents, e-mails, graphical engineering documents, images, videos, sensor data, and so on, that is, information is stored in so-called cross-media resources. Taking this situation into account, the next generation of semantic applications will have to address various challenges in order to come up with appropriate solutions, such as ontology learning and metadata generation, information integration and advanced ontology mapping. Whereas individual (non)logical approaches exist to address these aspects, one lacks a coherent framework to handle these challenges in an integrated way. Providing tools that still scale up, or designing the interaction with the users in such complex scenarios is still an open research issue.

References

- Alexaki, S., V. Christophides, G. Karvounarakis, D. Plexoudakis, and K. Tolle. 2001. The ICS-FORTH RDFSuite: Managing voluminous RDF description bases. 2nd International Workshop on the Semantic Web (SemWb2001), in conjunction with Tenth International World Wide Web Conference (WWW10), Hong Kong, 1–13.
- Alfonseca, E., and P. Rodriguez. 2002. Automatically generating hypermedia documents depending on user goals. Workshop on Document Compression and Synthesis in Adaptive Hypermedia Systems, Malaga, Spain.
- Angele, J., and Y. Sure, eds. 2002. Evaluation of ontology-based tools. In Proceedings of the 1st International Workshop EON2002. Siguenza, Spain. CEUR-WS.
- Aprirez, J.C., O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez. 2001. WebODE: A scalable workbench for ontological engineering. In Proceedings of the First International Conference on Knowledge Capture (K-CAP) 21–23 Oct 2001, Victoria, BC.
- Aumueller, D., H. Do, S. Massmann, and E. Rahm. 2005. Schema and ontology matching with coma++. In SIGMOD 2005 – Software Demonstration, Baltimore, MD.
- Benjamins, V.R., J. Contreras, M. Blazquez, J.M. Doderó, A. Garcia, E. Navas, F. Hernandez, and C. Wert. 2004. Cultural heritage and the semantic web. In *The Semantic Web: Research and Applications*, First European Semantic Web Consortium (ESWS2004), eds. C. Bussler, J. Davies, D. Fensel, and R. Studer, 433–444. Berlin: Springer.
- Bontcheva, K., H. Cunningham, A. Kiryakov, and V. Tablan. 2006. Semantic annotation and human language technology. In *Semantic web technologies: Trends and research in ontology-based systems*, eds. J. Davies, R. Studer, and P. Warren. Chichester, West Sussex: Wiley.

- Bouillon, Y., O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez. 2002. A survey on ontology tools. *OntoWeb: Ontology-based information exchange for knowledge management and electronic commerce*.
- Bozsak, E., M. Ehrig, S. Handschuh, A. Hotho, A. Madche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, and J. Tane, et al. 2002. KAON – Towards a large scale semantic web. In *Proceedings of the 3rd International Conference on E-Commerce and Web Technologies (EC-Web2002)*, eds. K. Bauknecht, A.M. Tjoa, and G. Quirchmayr, vol. 2455 of LNCS, 304–313. Aix-en-Provence, France: Springer.
- Broekstra, J., A. Kampman, and F. van Harmelen. 2002. Sesame: A generic architecture for storing and querying RDG and RDF schema. In *The 1st International Semantic Web Conference (ISWC2002)*, Sardinia, Italy.
- Buitelaar, P., P. Cimiano, and B. Magnini. 2005. *Ontology learning from text: Methods, applications and evaluation. Frontiers in artificial intelligence and applications*. Amsterdam: IOS Press.
- Buitelaar, P., D. Olejnik, and M. Sintek. 2004. A Protégé plug-in for ontology extraction from text based on linguistic analysis. In *Proceedings of the European Semantic Web Symposium (ESWS2004)*, Heraklion, Greece.
- Castano, S., A. Ferrara, and S. Montanelli. 2006. Matching ontologies in open networked systems: Techniques and applications. *Journal on Data Semantics* 5:25–63. Springer.
- Castano, S., A. Ferrara, S. Montanelli, G.N. Hess, and S. Bruno. 2007. State of the art on ontology coordination and matching. D4.4. Bootstrapping Ontology Evolution with Multimedia Information Extraction.
- Chakravarthy, A., F. Ciravegna, and V. Lanfranchi. 2006. Cross-media document annotation and enrichment. In *Proceedings of the 1st semantic web authoring and annotation workshop (SAAW2006)*, Athens.
- Chalupsky, H. 2000. OntoMorph: A translation system for symbolic knowledge. In *Proceedings of 7th International Conference on Knowledge Representation and Reasoning (KR)*, Breckenridge, CO, 471–482.
- Cheboto, A., L. Yu Deng, F. Fotouhi, A. Aristar, H. Brugman, A. Klassmann, H. Sloetjes, A. Russel, and P. Wittenburg. 2004. OntoELAN: An ontology-based linguistic multimedia annotator. In *Proceedings of IEEE 6th International Symposium on Multimedia Software Engineering*, Miami, 329–336.
- Cimiano, P., S. Handschuh, and S. Staab. 2004. Towards the self-annotating web. In *Proceedings of WWW2004*, New York, NY.
- Cimiano, P., and J. Volker. 2005. *Text2onto. In natural language processing and information (LNCS)*, 227–238. Berlin and Heidelberg: Springer.
- Ciravegna, F., A. Dingli, D. Petrelli, and Y. Wilks. 2002. Timely and non-intrusive active document annotation via adaptive information extraction. In *Semantic Authoring, Annotation and Knowledge Markup (SAAKM2002)*. ECAI, Lyon, France.
- Ciravegna, F., and Y. Wilks. 2003. Designing adaptive information extraction for the semantic web in Amilcare. In *Annotation for the semantic web*, eds. S. Handschuh, and S. Staab. Amsterdam: IOS Press.
- Corcho, O. 2004. A declarative approach to ontology translation with knowledge preservation. PhD thesis, Universidad Politecnica de Madrid.
- Corcho, O., A. Gomez-Perez, D.J. Guerrero-Rodriguez, D. Perez-Rey, A. Ruiz-Cristina, T. Sastre-Toral, and M.C. Suarez-Figueroa. 2003. Evaluation experiment of ontology tools' interoperability with the WebODE ontology engineering workbench. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Sanibel Island, FL.
- Cunningham, H., D. Maynard, V. Tablan, C. Ursu, and K. Bontcheva. 2002. Developing language processing components with GATE. Available at www.gate.ac.uk.
- De Leenheer, P., J. Kopecky, and E. Sharf. 2006. Requirements and design for an ontology versioning tool. *Data, Information and Process Integration with Semantic Web Services*. D2.4.

- Dill, S., N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingram, T. Kanungo, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien. 2003. SemTag and seeker: Bootstrapping the semantic web via automated semantic annotation. In Proceedings of WWW2003, Budapest, Hungary.
- Doan, A., J. Madhavan, P. Domingos, and A. Halevy. 2004. Ontology matching: A machine learning approach. In *Handbook on ontologies in information systems*, eds. S. Staab, and R. Studer, 397–416. Heidelberg: Springer.
- Domingue, J., M. Dzbór, and E. Motta. 2004. Magpie: Supporting browsing and navigation on the semantic web. In Proceedings ACM Conference on Intelligent User Interfaces (IUI), eds. N. Nunes, and C. Rich, 191–197. New York, NY: ACM Press.
- Dou, D., D. McDermott, and P. Qi. 2002. Ontology translation by ontology merging and automated reasoning. In Proceedings EKAW2002 Workshop on Ontologies for Multi-Agent Systems, Spain, 3–18.
- Duineveld, A.J., R. Stoter, M.R. Weiden, B. Kenepa, and V.R. Benjamins. 1999. Wondertools? A comparative study of ontological engineering tools. In Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW1999). Banff, Canada: Kluwer Academic Publishers.
- Ehrig, M., and S. Staab. 2004. QOM-quick ontology mapping. In Proceedings of the Third International Semantic Web Conference (ISWC2004), LNCS, eds. F. van Harmelen, S. McIlraith, and D. Plexoudakis, 683–696. Springer: Heidelberg.
- Euzenat, J., D. Loup, M. Touzani, and P. Valtchev. 2004. Ontology Alignment with OLA. In Proceedings of the 3rd EON Workshop, 3rd International Semantic Web Conference, Hiroshima.
- Euzenat, J., and L. Tardif. 2002. Xml transformation flow processing. *Markup Languages: Theory and Practice* 3(3):28–311.
- Feier, C., and J. Domingue. 2005. D3.lv0.l2 WSMO Primer, WSMO Working Draft.
- Fernandez-Lopez, M., A. Gomez-Perez, and M.D. Rojas Amaya. 2000. Ontology's crossed life cycles. In Proceedings of the 12th International Conference in Knowledge Engineering and Knowledge Management. Methods, models, and tools (EKAW2000), vol. 1937, 155–179, Juan-les-Pins, France: Springer.
- Fortuna, B., M. Grobelnik, and D. Mladenic. 2006. System for semi-automatic ontology construction. In Proceedings of the 3rd European Semantic Web Conference ESWC2006, Budva, Montenegro.
- Giunchiglia, F., and P. Shvaiko. 2004. Semantic matching. *The Knowledge Engineering Review* 18(3):265–280.
- Gomez-Perez, A., and M.C. Suarez-Figueroa. 2004. Evaluation of RDF(S) and DAML+OIL import/export services within ontology platforms. In Proceedings of the 3rd Mexican International Conference on Artificial Intelligence, Mexico, 109–118.
- Guarino, N., and C. Welty. 2002. An Overview of OntoClean. In *The handbook on ontology*, eds. S. Staab, and R. Studer, 151–172. Berlin: Springer.
- Jian, N., W. Hu, G. Cheng, and Y. Qu. 2005. Falcon-AO: Aligning ontologies with Falcon. In Proceedings of the K-CAP Workshop on Integrating Ontologies, Banff, Canada 85–91.
- Jones, S., and G.W. Paynter. 2002. Automatic extraction of document keyphrases for use in digital libraries: Evaluation and applications. *Journal of the American Society for Information Science and Technology (JASIST)* 53(8):653–677.
- Handschuh, S., S. Staab, and F. Ciravegna. 2002. S-CREAM – Semi-automatic CREATION of metadata. In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW2002), Siguenza, Spain, 358–372.
- Handschuh, S., S. Staab, and A. Maedche. 2001. CREAM – Creating relational metadata with a component-based, ontology-driven annotation framework. In First international conference on knowledge capture (KCAP2001), Victoria, Canada, eds. Y. Gil, M. Musen, and J. Shavlik, 76–83. New York: ACM Press. 1-581113-380-4.
- Heflin, J.D., and J.A. Hendler. 2000. Searching the web with SHOE. In Artificial Intelligence for Web Search, AAAI Workshop, Menlo Park, CA, 35–40.

- Kalfoglou, Y., and B. Hu. 2005. CMS: CROSI mapping system – Results of the 2005 ontology alignment contest. In Proceedings of the K-Cap'05 Integrating Ontologies Workshop, Canada, 77–84.
- Kalfoglou, Y., and M. Schorlemmer. 2003. If-map: An ontology-mapping method based on information-flow theory. *Journal on Data Semantics* 1:98–127.
- Kalyanpur, A., B. Parsia, E. Sirin, B. Cuenca-Grau, and J. Hendler. 2005. Swoop: A “web” ontology editing browser. *Journal of Web Semantics* 4(2): 144–153.
- Kerrigan, M. 2006. WSMOViz: An ontology visualization approach for WSMO. In 10th International Conference on Information Visualization, London.
- Kiryakov, A, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. 2005. Semantic annotation, indexing and retrieval. *Journal of Web Semantics* 2(1): 49–79.
- Klein, M., and D. Fensel. 2001. Ontology versioning for the semantic web. In Proceedings of the 1st International Semantic Web Working Symposium, 75–91. Stanford, CA: Stanford University.
- Kogut, P., and W. Holmes. 2001. AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. In 1st International Conference on Knowledge Capture (K-CAP2001). Workshop on Knowledge Markup and Semantic Annotation, Victoria.
- Kotis, K., and G. Vouros. 2003. Human-centered ontology management with HCONE. In Proceedings of Ontologies and Distributed Systems Workshop, IJCAI2003 Conference, Acapulco, Mexico.
- Kotis, K., G.A. Vouros, and K. Stergiou. 2006. Towards automatic merging of domain ontologies: The HCONE-merge approach. *Journal of Web Semantics* 4(1):60–79.
- Lambrix, P., M. Habbouche, and M. Perez. 2003. Evaluation of ontology development tools for bioinformatics. *Bioinformatics* 19(12):1564–1571.
- Maedche, A., B. Motik, N. Silva, and R. Volz. 2002. Mafra-a mapping framework for distributed ontologies. In Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW-2002, Madrid, Spain.
- Maedche, A., and S. Staab. 2001. Ontology learning for the semantic web. *IEEE Intelligent Systems, Special Issue on the Semantic Web* 16(2):72–79.
- Magkanaraki, A., G. Karvounarakis, T.T. Anh, V. Christophides, and D. Plexoudakis. 2002. Ontology storage and querying. Technical Report No.308. Foundation for Research and Technology Hellas, Institute of Computer Science, Information System Laboratory.
- McBride, B. 2001. Jena: Implementing the RDF model and syntax specification. In Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb2001), Hongkong.
- Modica, G., A. Gal, and H.M. Jamil. 2001. The use of machine-generated ontologies in dynamic information seeking. In The Proceedings of the 6th International Conference on Cooperative Information Systems, Trento, Italy, LNCS. Heidelberg: Springer.
- Niekrasz, J., and A. Gruenstein. 2006. NOMOS: A semantic web software framework for annotation of multimodal corpora. In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC2006), Genova, Italy.
- Noy, N., R. Ferguson, and M. Musen. 2000. *The knowledge model of Protégé-2000: Combining interoperability and flexibility of Lecture Notes in Artificial Intelligence (LNAI)*, vol. 1937, 17–32, Juan-les-Pins, France: Springer.
- Noy, N.F., and M.A. Musen. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. In Proceedings 17th National Conference on Artificial Intelligence (AAAI2000), Austin, TX.
- Noy, N.F., and M.A. Musen. 2002. PromptDiff: A fixed-point algorithm for comparing ontology versions. In 18th National Conference on Artificial Intelligence (AAAI2002). Edmonton, AB.
- Omelayenko, B. 2002. Integrating vocabularies: discovering and representing vocabulary maps. In Proceedings of 1st International Semantic Web Conference (ISWC2002), Chia Laguna (IT), 206–220.
- OntoWeb Consortium: A survey of ontology tools. IST Project IST-2000-29243, OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce, Deliverable 1.3, 2002.

- Rahm, E., and P.A. Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases* 10(4):334–350.
- SALERO Consortium. 2006. Representation techniques for multimedia objects. Deliverable 3.1.1.
- Schnurr, H.P., and J. Angele. 2005. Do not use this gear with a switching lever! Automotive industry experience with semantic guides. In 4th International Semantic Web Conference (ISWC2005), 1029–1040. Berlin and Heidelberg: Springer.
- Sintek, M., and S. Decker. 2001. TRIPLE-An RDF query, inference, and transformation language. In Proceedings of the Deductive Databases and Knowledge Management Workshop (DDL2001), Japan.
- Sleeman, D., and Q. Reul. 2006. CleanONTO: Evaluating taxonomic relationships in ontologies. In WWW2006, Edinburgh.
- Sofia Pinto, H., D.N. Peralta, and N.J. Mamede. 2002. Using Protege-2000 in reuse processes. In Proceedings of the International Workshop on Evaluation of Ontology-Based Tools. (EON2002), Siguenza, Spain.
- Stojanovic, L., and B. Motik. 2002. Ontology evolution within ontology editors. In Proceedings of the International Workshop on Evaluation of Ontology-Based Tools. (EON2002), Siguenza, Spain.
- Stumme, J., and A. Maedche. 2001. FCA-Merge: Bottom-up merging of ontologies. In 7th International Conference on Artificial Intelligence (IJCAI2001), Seattle, 225–230.
- Sure, Y., M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. 2002. OntoEdit: Collaborative ontology development for the semantic web. In Proceedings of the 1st International Semantic Web Conference: The Semantic Web (ISWC2002), Sardinia, Italy, eds. I. Horrocks, and J.A. Hendler, vol. 2342 of Lecture Notes in Computer Science (LNCS), 221–235. Heidelberg: Springer.
- Tuffield, M.M., S. Harris, D.P. Dupplaw, A. Chakravarthy, C. Brewster, N. Gibbins, K. O'Hara, F. Ciravegna, D. Sleeman, N.R. Shadbolt, and Y. Wilks. 2006. Image annotation with photocaption. In Proceedings of Semantic Web Annotation of Multimedia (SWAMM2006) Workshop at the World Wide Web Conference, Edinburgh.
- Vargas-Vera, M., E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. 2002. MnM: Ontology driven semi-automatic and automatic support for semantic markup. In 13th International Conference on Knowledge Engineering and Management (EKAW2002), eds. A. Gomez-Perez, V.R. Benjamins, 379–391. Heidelberg: Springer.
- Velardi, P., R. Navigli, A. Cuchiarelli, F. Neri. 2005. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In *Ontology learning from text: Methods, applications and evaluation*, eds. P. Buitelaar, P. Ciniamo, and B. Magnini. Amsterdam: IOS Press.
- Volkel, M., and T. Groza. 2006. SemVersion: An RDF-based ontology versioning system. In ICIW2006, Guadeloupe, France.
- Volker, J., D. Vrandečić, and Y. Sure. 2005. Automatic evaluation of ontologies (AEON). In Proceedings of the 4th International Semantic Web Conference, Galway, Ireland.
- Xu, Z., X. Cao, Y. Dong, and W. Su. 2004. Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies. In *Advances in knowledge discovery and data mining LNC*, 464–475. Berlin and Heidelberg: Springer.
- Zacharias, V., and S. Braun. 2007. SOBOLIO – Social Bookmarking and Lightweight Engineering of Ontologies. Workshop on Social and Collaborative Construction of Structured Knowledge (CKC), 16th International World Wide Web conference, (WWW2007), Banff, AB.

Chapter 7

Ontological Tools: Requirements, Design Issues and Perspectives

Konstantinos Kotis and George Vouros

An Ontological or Ontology Engineering tool is a piece of software that aims to support the engineering of ontologies. The engineering of ontologies concerns specific processes and tasks needed to be executed by domain experts and ontology engineers during the life-cycle of ontologies. The main objective of ontology engineering tools, especially in the context of the Semantic Web, is to provide certain functionality to users towards constructing *shared* and *evolving* ontologies. The aim of this article is to provide an understanding of the functionalities and technologies that need to be integrated in ontology engineering environments and to present issues that next generation ontological tools must consider in order to play a key role in the era of information and knowledge management.

7.1 Introduction

Ontologies explicate conceptualizations that are shaped and exploited by humans during practice and interaction among community members, binding the knowledge processes of creating, importing, capturing, retrieving, and using knowledge. Being part of knowledge that people possess, conceptualizations evolve in communities as part of “*knowing*” (Cook and Brown, 1999). “*Knowing*” is about interactions with the world as well with members of communities, with the aim to create new knowledge. Personal knowledge is created through practice, whereas group knowledge is created through interaction between community members.

The manipulation of conceptualizations involves their development, evaluation, exploitation and their continuous evolution as part of “*knowing*”, as humans perform their tasks and achieve their goals in their working contexts. In particular it involves:

K. Kotis (✉)
AI Lab, Department of Information and Communications Systems Engineering,
University of the Aegean, 83200 Karlovassi, Greece
e-mail: kotis@aegean.gr

1. *The development of personal conceptualizations.* People develop their own conceptualizations, which they may either make explicit (e.g. by formalizing terms in a special language, by taking notes about their meaning or just by naming concepts) or not (by storing them in the background of their minds). In their day-to-day activities people develop their conceptualizations either by improvising, by specializing/generalizing/aggregating existing concepts based on their experiences and on interaction with other community members, or by synthesizing existing conceptualizations.
2. *The development of commonly agreed group conceptualizations.* Developing commonly agreed and understandable domain conceptualizations is a very difficult and resource-demanding task that requires members of the communities to work synergistically towards shaping the information they exploit. By working synergistically, workers map others' conceptualizations to their own and put them in the context of their own experiences. This leads to a conversation whose back-and-forth, as it is pointed out in Cook and Brown (1999), not only results in exchanging knowledge but also in generating new knowledge.
3. *The evaluation and exploitation of conceptualizations.* Exploitation and evaluation of conceptualizations as part of the day-to-day practice of communities can be considered only as part of "knowing". Conceptualizations are put in practice and in the criticism of community members who, as already pointed out, have to compare them with their own conceptualizations and put them in the context of their own experiences (e.g. exploit them in a real working setting). Evaluation can result in new meanings, since concepts are seen under the light of new experiences and evolving contexts.

The above issues concerning the development, evaluation and exploitation of conceptualizations must be explicitly addressed in any ontology engineering methodology. The aim of an ontology engineering methodology is to prescribe processes and tasks that support humans to describe "what must be represented" and "how it can be represented" with respect to the conceptualization of domains. The objective of each ontology engineering methodology is to support the development and evolution of ontologies towards shared and agreed domain conceptualizations. Since the aim is on devising ontologies supporting their evolution towards commonly agreed conceptualizations, we emphasize collaborative methodologies that would support the active, decisive, and thus effective, participation of stakeholders in methodological tasks.

Therefore, in this article we discuss collaborative state-of-the-art ontology engineering (O.E) methodologies that imply important requirements for the functionality of O.E tools (Section 7.2). Based on the methodological implications of state-of-the-art collaborative engineering methodologies for evolving ontologies and on evaluation efforts for existing O.E tools, Section 7.3 presents a list of functional requirements that such tools must satisfy. Section 7.4 presents a table (in Appendix) that sums up known O.E tools' fulfilment of the stated requirements. Finally, Section 7.5 concludes this article with remarks and a research agenda of the issues that next generation ontological tools must consider in order to play their vital key role in this era of information and knowledge management.

7.2 The Engineering of Ontologies

The engineering of ontologies is the objective of several methodologies (Uschold and King, 1995; Grüninger and Fox, 1995; Fernández-López et al., 1999; Schnurr et al., 2000; Sure, 2002) that are analogous to the user-centered software engineering methodologies: Knowledge engineers with the participation of domain experts and/or ontology users gather the requirements for the ontologies, and either by using a knowledge representation language or by using an ontology engineering environment, they design and implement prototype ontologies. Ontologies produced are commented by experts in order to be further improved so as to reach a consensual conceptualisation of the corresponding domain. This contrasts to the old-fashioned technology-centered software engineering design approach in which developers gather requirements, design and implement an artefact and leave users to cope with what they have produced. However, even with the involvement of users, this approach to ontologies engineering is mostly machine-oriented: Knowledge engineers deal with these artefacts at the symbol level, mediating between domain conceptualisations and their formal representations, which can not be further manipulated or even (in some cases) be inspected by domain experts. This leads to a machine-oriented, knowledge engineer – centered ontology engineering approach: It relies heavily on de-contextualized principles of engineering ontologies using formal representation languages, but it does not deal with the way people manipulate their conceptualisations in the context of their day-to-day activities, individually or conversationally with colleagues. In order to (a) support the active and decisive involvement of knowledge workers in all stages of an ontology life-cycle, and (b) further empower people to decisively participate in the engineering of ontologies, shaping their information space in ways that are seamless to their day-to-day working activities, recent ontology engineering methodologies aim to accentuate the role of knowledge workers (i.e. domain experts, ontology users and knowledge engineers) and their active involvement in the ontology life-cycle (Pinto et al., 2004; Tempich et al., 2006; Kotis and Vouros, 2005).

In this section we provide detailed information on the latest O.E methodologies, HCOME (Kotis and Vouros, 2005) and DILIGENT (Tempich et al., 2006), which focus on the above themes with a greater respect than earlier methodologies. These modern approaches to the collaborative development of shared and evolving ontologies emphasize providing greater “opportunities” for knowledge workers to manipulate their conceptualizations during their day-to-day activities, and consequently they impose new functional requirements for the design of modern O.E tools.

In Kotis and Vouros (2005) authors conjecture that the most important issues that need to be considered within an ontology engineering methodology are:

1. *Allow an eclectic way for the development of ontologies.* Members of communities must be allowed to follow any approach or combination of approaches for the development of ontologies, which better fits their practice, their working norms and constraints: They may work in private by improvising conceptualizations and integrating concepts in a personal (i.e. not shared) conceptual system,

provide concepts with informal definitions, attaching information items to them, or/and choose to collaborate by comparing, merging and refining/generalizing existing ontologies.

2. *Emphasize the need for a “natural” (as much as possible) way to interact with conceptualizations.* A major issue for the decisive involvement of knowledge workers in any ontology engineering task is that they must interact with their conceptualizations at a level of detail that is more convenient for them. Therefore, low level implementation details of formal ontologies must be hidden from workers who may not understand knowledge representation formalisms’ terminology, syntax and semantics. People must be given the power to express subtle ontological distinctions in a way that is natural to them but satisfies the formal constraints of the specifications too. This further implies that knowledge workers must be supported to develop well-formed ontologies (i.e. ontologies without inconsistencies among the defined concepts, with coherency, and well organized), as well as to reuse, compare and map existing ontologies.
3. *Provide the means for exchanging, using and evaluating ontologies conversationally.* As already pointed out, shaping information synergistically is necessary, since knowledge is distributed among workers. To support conversations between individuals upon ontology specification, a methodology must enable further criticism on the developed artefacts, encourage feedback from community members, putting ontologies in the context of knowledge workers’ experiences and practice. Workers may deploy and evaluate ontologies during their day-to-day activities, raising arguments for and against specific conceptualization aspects, suggest changes, additions and refinements, and propose new ontology versions. Conversation facilities must provide the means for detecting new opportunities for collaboration, as well as for getting out of deadlocks within problematic situations that may arise during collaboration.
4. *Consider mapping of concepts’ definitions to other ontologies and/or lexical resources.* The aim is to uncover the human intended semantics (Uschold, 2003) of the specifications for clarification and communication purposes. This supports the bridging of different perspectives about the domain and provides a critical feedback on the precision of specifications. Mapping concepts according to their meaning is important for the development of commonly agreed conceptualizations, especially in communities where people from different disciplines use the same term with different meanings or use different terms for the same concept. Furthermore, for people to get a feedback on whether the developed specifications reflect the intended meaning of terms, or to further constrain the meaning of terms, they need to map the term definitions to word senses in a precise way.

The engineering of ontologies must be supported by O.E environments which are in accordance to the above methodological issues; i.e. integrated collections of tools/functionalities that support all (or most of the) phases of the modern collaborative ontology engineering methodologies.

7.2.1 The HCOME Methodology

The HCOME methodology (see Fig. 7.1) provides a clear distinction between the phases of the ontology life-cycle, between the goals that should be achieved in each phase and between the tasks that can be performed so as to achieve these goals. O.E tasks within the HCOME methodology are performed iteratively, until a consensus about a common ontology has been reached between knowledge workers. Tasks are performed by knowledge workers either individually or conversationally. In the first case, we consider that tasks are performed in a personal *space* of workers. In the

Ontology life-cycle phases	Goals	Tasks
Specification	Define aim / scope/ requirements/ teams	<ul style="list-style-type: none"> ▪ discuss requirements (S) ▪ produce documents (S) ▪ identify collaborators (S) ▪ specify the scope, aim of the ontology (S)
	Acquire knowledge	<ul style="list-style-type: none"> ▪ import from ontology libraries (P) ▪ consult generic top ontology (P) ▪ consult domain experts by discussion (S)
Conceptualisation	Develop & Maintain Ontology	<ul style="list-style-type: none"> ▪ improvise (P) ▪ manage conceptualisations (P) ▪ merge versions (P) ▪ compare own versions (P) ▪ generalize/specialize versions (P) ▪ add documentation (P)
	Use ontology	<ul style="list-style-type: none"> ▪ browse ontology (P) ▪ exploit in applications
Exploitation	Evaluate ontology	<ul style="list-style-type: none"> ▪ initiate arguments and criticism (S) ▪ compare others' versions (S) ▪ browse/exploit agreed ontologies (S) ▪ manage the recorded discussions upon an ontology (S) ▪ propose new ontology versions by incorporating suggested changes (S)

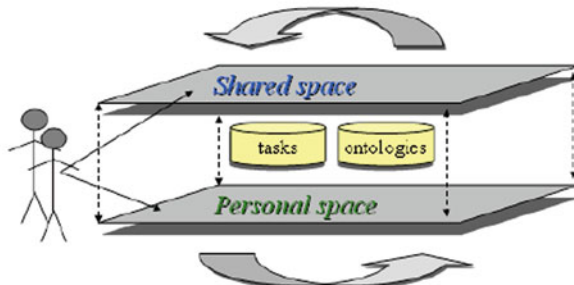


Fig. 7.1 The HCOME methodology: Phases/processes/tasks and the flow of tasks between stakeholders and spaces. The (S) symbol represents the shared space and (P) the personal space respectively

latter case, tasks are performed in an information space that a group of knowledge workers shares, i.e. in a *shared space*. A worker can initiate any ontology engineering task in his personal or shared space, or take part in a task that has been initiated by other members of the community in a shared space.

7.2.1.1 Specification Phase

During the HCOME specification phase, knowledge workers join groups that are about to develop shared ontologies. Having identified themselves within a group, during this initial phase of ontology engineering, workers are discussing requirements, producing specification documents, and agreeing on the aim and the scope of the new ontology. This phase may start from a worker that has already formed a conceptualisation of specific domain aspects and needs the contribution of colleagues to further develop the conceptualisation.

The “Specification” phase of the ontology life-cycle includes:

1. The specification of the scope and aim(s) of the ontology. This is essential in order for workers to have an agreed initial reference of the way they understand the domain and the way they intend to model it, according to their information needs.
2. An argumentation dialogue between the members of the team in order to obtain commonly agreed specification requirements.
3. Recording of the agreed specifications in appropriate forms and documents.

7.2.1.2 Conceptualization Phase

Having agreed on the scope and aim of the ontology to be developed, workers in their personal space can follow any approach or combination of approaches to the development of ontologies: They may improvise by integrating concepts in a conceptual system, provide concepts with informal definitions attaching information items to them, compare, merge and refine/generalize existing ontologies. Since the consultation of other well-known and/or widely acknowledged resources is critical to the ontology development process, knowledge workers may perform this task before sharing their conceptualisations with others. Collaborators should be able to create, store, maintain, compare, merge, and manage different versions of ontologies.

The “conceptualization” phase includes the following tasks:

1. The import of existing ontologies (for the reuse of conceptualisations. These may be located in collaborators’ personal spaces, in the shared space, in corporate internets, or in the World Wide Web.
2. The consultation of generic top ontologies, thesauruses and domain resources. The objective here is for knowledge workers to better understand and clarify the domain conceptualisations, and receive feedback concerning the informal, human intended semantics of the ontology specifications.

3. Improving to ontology construction by allowing the from-scratch development of formal ontologies based on workers' perception of the domain.
4. The mapping, merging and management of multiple versions of ontologies, supporting ontologies reuse and evolution.
5. The comparison of different versions of an ontology, for tracking ontology's evolution and for identifying ontologies that can possibly be merged.
6. Attaching to ontology elements information items, such as comments, examples and specification details.

7.2.1.3 Exploitation Phase

The need to reach a common conceptualization of the working domain, push inevitably ontologies developed in personal spaces to the shared space. As already said, shared ontologies can be used by knowledge workers in the context of specific ontology-driven applications and tasks, be exploited and be evaluated conversationally. The exploitation of an ontology version that has been developed by a colleague is seen as part of the ontology life-cycle since it may result to the provision of feedback for the conceptualizations developed, followed by a conversation between parties and to an agreement to a specific (new) conceptualization. The evaluation and further development of personal ontologies is achieved via a structured conversation and criticism upon the ontology versions posted in the shared space. The recording of this conversation enables the tracking of changes and of the rationale behind ontology versions, supporting the decisions on conceptualising the domain in the final ontology

The "Exploitation" phase includes:

1. The inspection of agreed or shared ontologies, either by individuals in their personal space or by collaborators in the shared space, for reviewing, evaluating and criticizing the specified conceptualisations.
2. The comparison of (personal and shared) versions of an ontology, for identifying the differences between them.
3. The posting of arguments upon versions of ontologies for supporting workers' decisions, for or against specifications.

7.2.2 The DILIGENT Methodology

The aim of the DILIGENT methodology is to support domain experts within a distributed working environment to design, develop and evolve ontologies following an argumentation approach based on Rhetorical Structure Theory (RST). RST (Mann and Thompson, 1987) has been used for the analysis of discussions towards reaching a consensus within evolving and distributed processes of ontology engineering. Based on case studies performed in the context of the DILIGENT methodology, one can draw the conclusion that argumentation frameworks for the collaborative engineering of ontologies contributes significantly to the acceleration

of the development of a commonly agreed ontology. The DILIGENT methodology integrates the following tasks:

1. Collaborative building of a “starting” shared ontology (O_1), by domain experts (K-provider), ontology engineers (O-engineer), knowledge engineers (K-engineer) and end-users (O-user)
2. Local adaptation of the shared ontology from the end-users and its exploitation in their personal space
3. Analysis of local ontologies (O_n) and of user-specific requirements by a control board, and decision for the changes/corrections that will be incorporated in the next version of the shared ontology
4. Revision and update of the shared ontology given the output of task 3
5. Local update of the local users’ ontologies, based on the new version of the updated shared ontology (Fig. 7.2)

The process starts with the development of a starting ontology, the so called “shared” ontology, with the collaboration of domain experts, ontology engineers, knowledge engineers and end-users. The aim is to involve different people, in different roles, with different needs and goals that may not be in the same place. Since the starting ontology is available, end-users may use it and adapt it in their own local needs: At their local personal space, end-users are allowed to change the shared ontology, however they cannot change the ontology that resides in the shared space. A “control board” is responsible for updating the shared ontology by gathering and analyzing end-users’ specifications for changes. The responsibility of the control board is to keep a balance of the degree of each participant’s participation and reconcile conflicts in specifications, ensuring that the shared ontology does not differ much from the local ontologies. On the other hand, end-users must update their local ontology in order to get the latest version of the shared conceptualization.

In contrast to the traditional methodologies [e.g. METHONTOLOGY (Fernández-López et al., 1999), ON-TO-KNOWLEDGE (Schnurr, 2000; Sure, 2002)] to ontology engineering, the DILIGENT methodology is very close to HCOME. We would say that both efforts move towards the third-generation of ontology engineering methodologies. Specifically, both methodologies consider distributed settings and thus emphasize issues concerning collaboration and

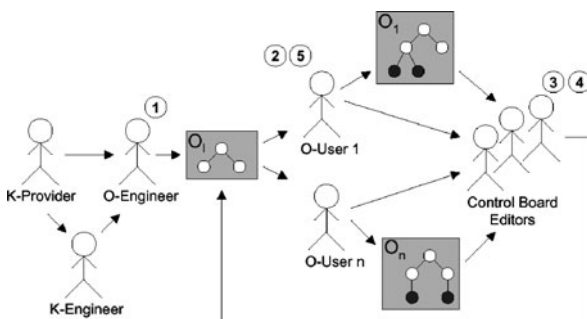


Fig. 7.2 The DILIGENT methodology (Tempich et al., 2006)

argumentation. In addition, both methodologies consider evolving ontologies, pointing on the importance of argumentation dialogues, version management and merging of ontologies. Finally, they both consider that engineers must somehow minimize their involvement during the engineering of ontologies, at least as far as the specification of the ontologies is concerned, accentuating the role of domain experts and ontology users. However the two approaches tackle this last issue differently: DILIGENT is clearly dependent on decisions of a control board, whereas HCOME aims to empower knowledge workers to actively and decisively participate in the ontology life-cycle possibly without the assistance of knowledge/ontology engineers.

7.3 Next-Generation Ontology Engineering Tools

Several related reports on ontology engineering tools [e.g. (Garcia-Castro, 2004; Gómez Pérez, 2002)] establish a set of criteria for evaluating them, reporting on advantages and disadvantages of their design. These reviews can be seen as a kind of analysis of functional requirements that one may (partially) adopt for designing new tools; i.e tools that consider the pros and cons of existing implementations, providing the necessary functionality for supporting the processes and tasks of state-of-the-art O.E methodologies. Based on this bibliography and on the methodological implications of HCOME and DILIGENT methodologies described in Section 7.2, we have concluded on a list of important requirements for designing modern O.E tools. Section 7.4 provides information on the requirements fulfilled by existing O.E tools.

1. *Edit/View*: The most common functionality of O.E tools is the editing of ontologies. At a higher level, this functionality may be performed using many different interface technologies: (a) forms i.e. template-driven definition of classes/properties/instances using a knowledge representation language (b) natural language definition based on a controlled language or on the WYSIWYM knowledge editing paradigm (c) graphical representation of definitions (d) hypertext linkage of classes/properties/instances supporting definitions' navigation and (e) a combination of such technologies. At a lower level, this task requires the use of a formal language in order to formally represent the specified conceptualizations. Several ontology-specific languages have been proposed, with varying degrees of formality. Currently, the Web Ontology Language (OWL) family of languages is the leading standard for the specification of ontologies. Concluding the above, we require that a modern O.E tool provides users with a synthesis of different ways to edit and view ontologies. More important, tools must provide facilities that empower end-users with no experience in ontology engineering and knowledge representation to manipulate their conceptualizations in the most "natural" way for them, maintaining the consistency of specifications.

2. *Reasoning*: Modern O.E tools should provide the functionality to (a) reason with domain knowledge, inferring new knowledge from the knowledge base (b) check the consistency of specifications (Garcia-Castro, 2004; Gómez Pérez, 2002). Reasoning services must perform in the background, providing sufficient information to people about the well formed-ness of their ontologies. Feedback from the reasoning services must be provided in the form of help and advice about the validity and consistency of specifications, with respect to people cognitive capacities (i.e. avoid feedback messages in formal language and/or using technical terms related to the representation and reasoning system).
3. *Collaboration*: Modern O.E tools must support the collaborative development of shared ontologies. Tools must provide access to the collaborative and concurrent engineering of multiple ontologies by multiple users. We specifically identify two design requirements:
 - a. Tools must be able to distinctively provide at least two different information spaces: The personal and the shared space. This means that, at a given time point, an ontology should be characterized to be personal or shared according to how many users are allowed to participate to its development. Being in the personal space, ontologies are developed by only one user, whereas shared ontologies are developed collaboratively by more than one user at the same time. The management of ontologies within these two different spaces does not necessarily imply the creation of two physically distinct spaces. A shared ontology that is finally agreed between the members of a collaborative team of developers is considered to be a commonly agreed ontology, and can be optionally stored to another (virtual) space i.e. to the “agreed” space.
 - b. Tools must support the organization of discussion sessions for the exchange of arguments upon shared ontologies. These sessions allow a collaborative development of shared ontologies using the power of argumentation models. When joining such a session, knowledge workers agree to share a particular ontology. The aim and scope of the shared ontology must also be collaboratively defined and be agreed. Arguments concerning users’ agreement or disagreement on specific shared conceptualizations must be recorded, together with new suggested conceptualizations, in the form of new versions of the shared ontology. To better track the discussion, argumentation dialogues may follow an argumentation model for the specification of the types of argumentation items and their relations.
4. *Support for multiple ontologies*: O.E tools need to support the concurrent management of multiple ontologies. Tools must provide mechanisms to create several views (versions) of an ontology and to manage different ontologies within the same context. Users must be capable of managing more than one ontologies at a time. We specifically identify the following requirements:
 - a. Tools must be able to load more than one ontologies at the same time for users to be able to inspect, consult and manipulate them in parallel. For

- instance, parts of an ontology must be allowed to be copied to another one in a consistent way. Also, the alignment and merging of two ontologies must be supported.
- b. Tools must allow the creation of several views (versions) of an ontology. They must support versioning in a very effective way. Versions of an ontology must be organized based on several criteria, e.g. by the date of creation, the contributing party, the version number, etc. Change operations between subsequent versions must be recorded separately.
 - c. To control the evolution of ontologies effectively, rich meta-information must be related to ontology versions: When a version of an ontology is created during a collaborative session (argumentation), the argumentation items placed must be interrelated with the corresponding changes (if any) for future exploitation (this is further elaborated in point 5 below).
 - d. Tools must support the comparison of different ontologies. This comparison can be based on the changes recorded during the evolution of an ontology (in the case of two versions of an ontology), or it can be based on comparison mechanisms for computing differences (syntactic, semantic, structural) between two ontologies: For instance, this can be an ontology alignment algorithm (in the best case) or a simple syntactic matching algorithm (in the simplest case).
5. *Reuse of knowledge*: Ontology engineering tools manage knowledge about the domain of discourse (represented by a domain ontology), but also information about this knowledge (e.g. administrative information, information about versions and changes that have occurred during ontology evolution, argumentation dialogues). The exchange of ontologies between and within tools must not be restricted to the level of domain knowledge but must be extended to the level of meta-information. For the specification of the information to be exchanged, we conjecture that the use of formal models is the right solution: The ontology engineering process itself involves knowledge-intensive activities performed by members of specific communities. People participating in such a process need to share a common understanding of the various aspects and issues involved (e.g. domain, methodological and tool-related ones). Therefore (meta-)ontologies¹ can play a major role in interlinking, sharing and combining knowledge among the parties involved in a collaborative ontology engineering process, either within the same environment (exploiting conceptualizations at the shared space) or across different O.E environments (Kotis et al., 2007). We specifically identify three major design implications:
- a. Tools must provide the mechanisms for importing/exporting and relating meta-information to any domain ontology. The attachment of such knowledge must be performed by recording instances of the (meta-)

¹To distinguish among domain ontologies and ontologies concerning types of meta-information, when this is needed, we refer to the former as “domain ontologies” and to the latter as “(meta-)ontologies”.

- ontologies. When a user fetches an ontology version from an ontology store, he must retrieve also all the related meta-information: the identity of the previous version, the change operations that have occurred since the last version, the argumentation items related to these changes, administrative meta-information. This will allow him to inspect the evolution history and decide on the exact contributions he has to make.
- b. Meta-information must be captured modularly and independently of the domain ontology, and must be represented using standard semantic web languages for interoperability purposes. A recent effort towards this new design implication is presented (both at the requirements and specification as well as at the implementation level) in an extended work of HCOME methodology, called the HCOME-30 framework (Vouros et al., 2007).
 - c. Tools must be able to exchange meta-information, thus there is a need for exploiting a common vocabulary, i.e. agreed (meta-) ontologies. In case different ontology engineering tools exploit different (meta-) ontologies, there must be an alignment mechanism of (meta-) ontologies so as meta-information to be readily aligned.
6. *Consultation*: The development of ontologies must be supported by a consultation functionality which is usually integrated in O.E tools in the following forms: (a) capability to inspect generic semantic distinctions of general / upper ontologies such as SUMO (Pease et al., 2002) (b) capability of browsing general purpose lexicons or thesaurus such as WordNet (Miller, 1995) in order to obtain the meanings (senses) of classes/properties (or automatically assigning these senses) (c) getting advice by experts or control boards during an argumentation dialogue, and (d) getting feedback from a reasoner, mainly concerning the consistency of definitions.
 7. *Querying*: This concerns the support for the formation of queries to the knowledge base, driven by the definition of classes/properties. Reasoning facilities must support this functionality for deducing new knowledge from the existing one.
 8. *Extensibility*: The design of O.E tools should provide support so as their functionality to be extended in a rather easy fashion. Dynamic and continuously evolving settings such as the Semantic Web must host applications that are adapted to changes fast and easily, by allowing the efficient integration of new functionalities. The design of tools based on extensible software platforms (e.g. plug-ins or web services) ensures easy maintenance and high usability.
 9. *Open to services*: O.E tools must provide any ontology engineering functionality as a Web service (Dameron et al., 2004). Web services are considered a key technology for the World Wide Web. The ability to discover and use ontology engineering services provided by others, at any time, from any place, and from any platform, empowers users to develop ontologies on a larger scale. The potential of web services opens a new era to the design and development of O.E platforms.
 10. *Scalability/Storage*: The ability of O.E tools to manage large ontologies is a requirement placed by real Semantic Web applications (Motta, 2006). This is in contrast to case studies and prototype developments (Garcia-Castro, 2004;

Gómez Pérez, 2002). Modern O.E tools must facilitate the persistent storage of domain knowledge using powerful database management systems.

11. *Interoperability*: The design of O.E tools must provide a wide range of translation mechanisms in order to support the import/export of different ontologies at the level of formal languages. From RDF(S) to OWL (and vice versa), modern tools must support the translation of an imported/exported ontology with the minimum loss in expressiveness.
12. *Open to non-semantic resources (Ontology learning)*: An alternative way to develop a domain ontology is by (semi)automatically discovering and extracting concepts and their relations from text corpora, dictionaries, knowledge bases or other schemata/ontologies. To carry out this task automatically, natural language analysis and machine learning techniques must be integrated. The design of ontology engineering tools must provide the ability to extract knowledge from arbitrary sources in the conventional Web (Motta, 2006) and automatically build domain ontologies to support further annotation.

7.4 Supporting Ontology Engineering

In this section we examine whether existing O.E tools support the engineering of ontologies by satisfying the above requirements. We present O.E tools that are integrating more than one functionalities to support (fully or partially) a collaborative O.E methodology. These O.E tools can be considered as “*integrated ontology engineering (O.E) environments*”. In conjunction we present “*self-standing O.E tools*”, which are considered to support only a specific O.E functionality. For instance, these can be tools that provide information on differences between ontology versions.

7.4.1 Integrated O.E Environments

Table 7.1 in Appendix summarizes the degree of functional support that integrated O.E environments currently provide. The functionalities/criteria of the table are drawn from the requirements presented in Section 7.3. The O.E environments selected for this presentation are the most well known and widely used/published. This table can be seen only as a report on the existence of tools’ functionality and not as a competition for the best tool. Although tools that fully support a collaborative O.E methodology should be considered as the most complete, other tools that partially support stated requirements may be suitable for specific ontology engineering tasks for knowledge workers that, in a specific context, have limited demands on O.E tools’ functionality.

As a conclusion that can be drawn from this table, the HCONE integrated O.E environment provides the functionality needed for the HCOME collaborative O.E methodology, since it was designed based on the requirements of the methodology. SWOOP and Protégé, although missing integrated functionalities to support collaboration, they are positioned in the higher place of the most popular

and widely used freely downloadable O.E tools because of the extensive range of functionalities they integrate and their plug-in architecture.

7.4.2 *Self-Standing O.E Tools*

O.E tools, as already stated, are tools that support only a specific O.E functionality; e.g. tools for providing information on differences between ontology versions or tools for supporting only the editing of ontologies. In the following we provide examples of such tools, categorized according to the functionality that they support:

1. *Edit/view*: In contrast to integrated ontology environments that provide support for editing and viewing ontologies in conjunction to other provided functionalities, tools such as OilEd (Bechhofer et al., 2001) provide support for (a) the editing of ontologies (b) the viewing of ontological definitions using a single viewing facility and (c) consistency checking. These tools cannot be evaluated as integrated ontology environments since they support only a very specific task of an O.E methodology and they are not designed so as to support other important issues such as scalability, extensibility, collaboration, argumentation, and reuse of ontologies.
2. *Alignment tools*: Ontology alignment is the task of establishing a collection of binary relations between the vocabularies of two ontologies, with respect to the original axioms. The mapping between two ontologies can be defined as a morphism from one ontology to the other, i.e. as a function that establishes equivalence relations between concepts and properties of ontologies (Kotis and Vouros, 2004). The alignment/mapping of ontologies is a necessary step towards merging ontologies. Most of the well-known ontology alignment tools are single-functionality tools that support mainly the OWL language [e.g. AUTOMS (Kotis et al., 2006), FALCON-AO (Hu et al., 2006), COMA++ (Massmann et al., 2006)] and provide results in an automatic way. There are also ontology alignment plug-ins integrated in ontology environments [e.g. Prompt (Noy and Musen, 2000) in Protégé, HCONE-merge (Kotis and Vouros, 2004) in HCONE, ODEmerge (Arpírez et al., 2001) in WebODE].
3. *Learning*: According to the source from which knowledge is extracted, ontology learning tools are categorized as follows: (a) from text [e.g. Text-To-Onto (Meadche and Staab, 2004)] (b) from dictionaries [e.g. DODDLE (Gómez Pérez and Manzano-Macho, 2003)] (c) from knowledge bases (d) from schemata [e.g. OntoBuilder (Gómez Pérez and Manzano-Macho, 2003)]. Furthermore, there are also tools that can be “plugged” in integrated environments [e.g. OntoLT in Protégé (Buitelaar et al., 2004)].
4. *Versioning*: This is more an integrated functionality of ontology engineering environments or a plug-in functionality in ontology tools, such as PromptViz plug-in in Protégé 3.x. However, there are also tools that provide support only for this functionality. An example is OntoView (Klein et al., 2002), which provides support for comparing two versions of an ontology by returning their differences.

5. *Evaluation*: Evaluation concerns (a) the evaluation of the ontologies developed by tools and (b) the evaluation of tools that develop ontologies. The criteria for both types of evaluation can be based on evaluation frameworks such as the one proposed by OntoWeb project (Gómez Pérez, 2002). According to this framework, ontologies are evaluated for their “language conformity” (i.e. to the syntax of a representation language) and for their “consistency” (i.e. to what extent they ensure consistency of specifications with respect to their semantics), whereas tools are evaluated for their technological properties such as interoperability, performance, memory allocation, scalability, interface. Examples of evaluation tools are: OntoAnalyser (Gómez Pérez, 2002) (ontology evaluation), OntoGenerator (Gómez Pérez, 2002) (tools evaluation), OntoClean (Gómez Pérez, 2002) (ontology evaluation), One-T (Gómez Pérez, 2002) (ontology evaluation). Having said that, we must distinguish these types of evaluation from the ontology exploitation phase supported, for instance, by the argumentation and ontology comparison functionality of collaborative O.E environments.

7.5 Conclusion

In this article we have presented an overview concerning efforts for the engineering of ontologies using state-of-the-art methodologies and tools. We have shown that the design of such tools should be driven by specific implementation requirements and methodological issues that latest O.E methodologies such as HCOME or DILIGENT explicitly specify. A list of important issues concerning the design of modern O.E tools has been outlined. Based on this list, we accentuate the importance of the following points:

1. The role of a collaborative engineering methodology to the design of O.E tools is rather significant and should be always considered when O.E functionalities for the development of ontologies are chosen.
2. Tools must support the exchange of domain conceptualizations between communities of users by sharing rich meta-information about the conceptualizations developed. This information, shaped by (meta-)ontologies, can play a major role in the interlinking, sharing and combination of knowledge among the parties involved in a collaborative ontology engineering process.
3. The use of (Semantic) Web Services towards the trading of O.E functionality in a global market of O.E services opens new horizons to the engineering of ontologies, to the realization of O.E tools and to the realization of the Semantic Web.

Keeping in mind the above points, the engineering of ontologies, driven by the requirements of continuously evolving, distributed and open application environments such as the Semantic Web, could get new dimensions and meaning if it is viewed from a knowledge-worker-centered angle: We can create the opportunities for millions of web users and developers to actively play their role in the development and deployment of ontologies in a way that is seamless to their everyday tasks. This is a challenge for the evolution of the Semantic Web.

Appendix

Table 7.1 Functionality support of integrated O.E environments

	HCONe	WebODE	OntoStudio	KAON	Protégé	SWOOP	WebOnto
Collab. methodology support	✓Full	✓Partial	✓Partial	✓Partial	✓Partial	✓Partial	✓Partial
Edit/view	No	✓	✓	✓	✓	✓	No
Form-based	✓	No	No	No	No	Future	No
Natural language	✓	✓	Partial	✓	✓	Future	✓
Graph-based	✓	No	No	No	✓	✓	No
Formal (OWL)	✓	✓	No	No	No	✓	No
HyperText-driven	✓	✓	No	No	No	✓	No
Reasoning services	✓	✓	✓	✓	✓	✓	✓
Collaboration	✓	✓	No	No	✓	✓	✓
Personal/shared ontologies	✓	✓	No	No	✓	✓	✓
Argumentation/dialogues	✓	No	No	No	No	No	✓
Support for multiple ontologies	✓/Designed	✓/No	✓(Mapping)/No	✓(Inclusion)/No	✓/No	No/No	No/No
Merge ontologies/partitions	✓	No	No	✓	✓	✓	No
Changes management	✓	No	No	✓	✓	✓	No
Versions management/comparison	✓	No	No	No	✓	No	No

Table 7.1 (continued)

	HONE	WebODE	OntoStudio	KAON	Protégé	SWOOP	WebOnto
Re-usability	Meta-information Representation of conceptualizations	Future ✓(OWL)	No ✓(OWL, RDF(S), F-logic)	No ✓(OWL, RDF(S), F-logic)	No ✓(OWL, RDF(S), DAML+ OIL, UML)	No ✓	No No
Consultation	Lexicon-driven Upper ontology driven Experts/board driven	✓ No No No	No No No	✓ No No	✓ No No	No No No	No No ✓
Querying KB		Designed	✓	✓	✓	✓	✓
Extensibility		No	No	No	✓	✓	No
Scalability/DB storage		✓	✓	✓	✓	No	No
Ontology learning		No	No	✓	No	No	No
Services implementation		No	No	No	No	No	No

References

- Arpírez, J.C., O. Corcho, M. Fernández-López, and A. Gómez-Pérez. 2001. WebODE: A scalable workbench for ontological Engineering. First International Conference on Knowledge Capture (KCAP'01), 6–13. New York: ACM Press (1-58113-380-4).
- Bechhofer, S.I., I. Horrocks, C. Goble, and R. Stevens. 2001. OilEd: A Reason-able ontology editor for the semantic web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, 19–21 Sept, Vienna. LNAI, Vol. 2174, 396–408. Heidelberg: Springer.
- Bernstein, A., and E. Kaufmann. 2006. GINO – A guided input natural language ontology editor. 5th International Semantic Web Conference (ISWC 2006), Heidelberg: Springer.
- Buitelaar, P., D. Olejnik, and M. Sintek. 2004. A protégé plug-in for ontology extraction from text based on linguistic analysis. In 1st European Semantic Web Symposium, ESWS-04, Heidelberg: Springer.
- Cook, S.D.N., and J.S. Brown. 1999. Bridging epistemologies: The generative dance between organizational knowledge and organizational knowing. *Organizational Science* 10:381–400.
- Dameron, O., N.F. Noy, H. Knublauch, and M.A. Musen. 2004. Accessing and manipulating ontologies using web services. In Workshop of “Semantic Web Services: Preparing to Meet the World of Business Applications”, 3rd International Semantic Web Conference (ISWC 2004), Heidelberg: Springer.
- Domingue, J. 1998. Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web. 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, 18th–23rd April. Banff, AB.
- Fernández-López, M., A. Gómez-Pérez., A. Pazos-Sierra, and J. Pazos-Sierra. 1999. Building a chemical ontology using METHONTOLOGY and the ontology design environment. *IEEE Intelligent Systems & their applications*, 37–46.
- García-Castro, R. 2004. Specification of a methodology, general criteria, and benchmark suites for benchmarking ontology tools. EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB, Deliverable D2.1.4.
- Gómez Pérez, A. 2002. A survey on ontology tools. OntoWeb: Ontology-based information exchange for knowledge management and electronic commerce. IST-2000-29243 Deliverable 1.3.
- Gómez Pérez, A., and D. Manzano-Macho. 2003. A survey of ontology learning methods and techniques. OntoWeb: Ontology-based information exchange for knowledge management and electronic commerce. IST-2000-29243 Deliverable 1.3.
- Grüninger, M., and M.S. Fox. 1995. Methodology for the design and evaluation of ontologies. Proceedings of IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, QC.
- Hoffman, R., P. Hays, and K. Ford. 2002. The triples rule. *IEEE Intelligent Systems*, 62–65.
- Hu, W., G. Cheng, D. Zheng, X. Zhong, and Y. Qu. 2006. The results of Falcon-AO in the OAEI 2006 Campaign. OAEI (Ontology Alignment Evaluation Initiative) 2006 contest, Ontology Matching International Workshop, Atlanta, GA.
- Klein, M., D. Fensel, A. Kiryakov, and D. Ognyanov. 2002. Ontology versioning and change detection on the web. *EKAUW 2002*, 197–212.
- Kotis, K., and G. Vouros. 2003. Human centered ontology management with HCONE. In proceedings of ODS Workshop, IJCAI-03, Acapulco.
- Kotis, K., and G. Vouros. 2004. HCONE approach to ontology merging. *The semantic web: Research and applications, first european semantic web symposium, ESWS'04, Heraklion, Crete, Greece*, eds. J. Davies, D. Fensel, C. Bussler, and R. Studer, 10–12 May, Proceedings, Series: Lecture Notes in Computer Science, Vol. 3053, Heidelberg: Springer.
- Kotis, K., and G. Vouros. 2005. Human-Centered Ontology Engineering: The HCOME methodology. *International journal of knowledge and information systems (KAIS)*, Springer, Online First.

- Kotis, K., A. Valarakos, and G. Vouros. 2006. AUTOMS: Automating ontology mapping through synthesis of methods. OAEI (Ontology Alignment Evaluation Initiative) 2006 contest, Ontology Matching International Workshop, Atlanta, GA.
- Maedche, A., and S. Staab. 2004. Ontology learning. In *Handbook on ontologies*, eds. S. Staab, and R. Studer, Springer.
- Mann, W., and S. Thompson. 1987. Rhetorical structure theory: A theory of text organization. In *The structure of discourse*, ed. L. Polanyi, Ablex Publishing Corporation: Norwood, NJ.
- Massmann, S., D. Engmann, and E. Rahm. 2006. COMA++: Results for the ontology alignment contest OAEI 2006. OAEI (Ontology Alignment Evaluation Initiative) 2006 contest, Ontology Matching International Workshop, Atlanta, GA.
- Miller A.G. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38(1):39–41.
- Motta, E. 2006. Next-generation semantic web applications. In 1st Asian Semantic Web Conference, China. Heidelberg: Springer.
- Noy, N., and M. Musen. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. In the Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX. Available as SMI technical report SMI-2000-0831.
- Pease, A., I. Niles, and J. Li. 2002. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web, Edmonton, Canada, 28 Aug – 1 July.
- Pinto, H.S., S. Staab, Y. Sure, and C. Tempich. 2004. OntoEdit empowering SWAP: A case study in supporting distributed, loosely-controlled and evolving engineering of ontologies (DILIGENT). In 1st European Semantic Web Symposium, ESWS 2004, Heidelberg: Springer.
- Pinto, S., and J. Martins. 2004. Ontologies: How can they be built? *Knowledge and Information Systems* 6:441–464.
- Schnurr, H.P., Y. Sure, and R. Studer. 2000. On-to-knowledge methodology – Baseline version. Executive summary, On-To-Knowledge EU-IST-1999-10132 Project Deliverable D15.
- Staab, S., R. Studer, H. Schnurr, and Y. Sure. 2001. Knowledge processes and ontologies. *IEEE Intelligent Systems* 16(1):26–34.
- Sure, Y. 2002. A Tool-supported methodology for ontology-based knowledge management. ISMIS 2002, Methodologies for Intelligent Systems.
- Tempich, C., H.S. Pinto, and S. Staab. 2006. Ontology engineering revisited: An iterative case study. In 3rd European Semantic Web Conference, ESWC 2004, Heidelberg: Springer.
- Uschold, M. 2003. Where are the semantics in the semantic web? *AI Magazine* 24:25–36.
- Uschold, M., and M. King. 1995. Towards a methodology for building ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, QC.
- Vouros, A.G., K. Kotis, C. Chalkiopoulos, and N. Lelli. 2007. The HCOME-3O framework for supporting the collaborative engineering of evolving ontologies. ESOE 2007 International Workshop on Emergent Semantics and Ontology Evolution, ISWC 2007, 12 Nov Busan, Korea.

Chapter 8

Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages

Giancarlo Guizzardi and Gerd Wagner

8.1 Introduction

In recent years, there has been a growing interest in the use of foundational ontologies (also known as upper level, or top-level ontologies) for: (i) evaluating conceptual modeling languages; (ii) developing guidelines for their use; (iii) providing real-world semantics for their modeling constructs. In this paper, we present a fragment of a philosophically and cognitively well-founded reference ontology named *UFO (Unified Foundational Ontology)*. UFO started as a unification of the GFO (Generalized Formalized Ontology; Heller and Herre, 2004) and the Top-Level ontology of universals underlying OntoClean (<http://www.ontoclean.org>). However, as shown in Guizzardi (2005), there are a number of problematic issues related the specific objective of developing ontological foundations for general conceptual modeling languages (e.g., EER, UML, ORM) which were not covered in a satisfactory manner by existing foundational ontologies such as GFO, DOLCE or OntoClean. For this reason, UFO has been developed into a full-blown reference ontology of endurants based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. This ontology is presented in depth and formally characterized in Guizzardi (2005). In Section 8.2, we discuss the main categories comprising UFO.

Furthermore, we demonstrate in this paper how this ontology can be used in the design and evaluation of conceptual modeling languages. In Section 8.3, we present a general ontology-based framework that can be used to systematically assess the suitability of an artificial modeling language to model phenomena in a given domain. In particular, this framework focuses on two properties of modeling languages (Guizzardi, 2005): (i) *domain appropriateness*, which refers to truthfulness of a language to a given domain in reality; (ii) *comprehensibility appropriateness*, which refers to the pragmatic efficiency of a language to support communication, domain understanding and reasoning in that domain.

G. Guizzardi (✉)

Federal University of Espirito Santo (UFES), Vitoria-ES, Brazil; Laboratory for Applied Ontology (ISTC-CNR), Trento, Italy

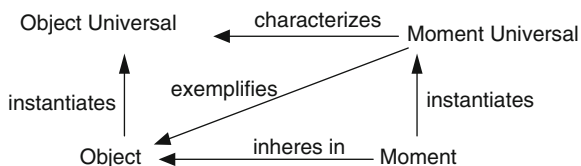
e-mail: gguizzardi@inf.ufes.br

In Sections 8.4 and 8.5, we employ UFO and the framework of Section 8.3 to analyze and redesign the 2.0 version of the metamodel of the Unified Modeling Language (UML; Object Management Group, 2003). The fact that UML is a *de facto* standard considered in several sub-fields of computer science (e.g., software and domain engineering, database and information systems design) counts in favor of the practicality and relevance of this approach. Section 8.6 presents some final considerations of the article.

8.2 The Unified Foundational Ontology (UFO)

In the sequel, we restrict ourselves to a fragment of UFO, depicted in the Fig. 8.1. Moreover, due to space limitations and the focus of the paper we present the ontological categories comprising UFO superficially. For an in depth presentation and corresponding formalization, one should refer to Guizzardi (2005).

Fig. 8.1 The Aristotelian square



8.2.1 The Core Categories: Object–Object Universal, Moment–Moment Universal

A fundamental distinction in this ontology is between the categories of *Particular* and *Universal*. Particulars are entities that exist in reality possessing a unique identity. Universals, conversely, are pattern of features, which can be realized in a number of different particulars. The core of this ontology exemplifies the so-called *Aristotelian ontological square* or what is termed a “*Four-Category Ontology*” (Lowe, 2006) comprising the category pairs *Object–Object Universal*, *Moment–Moment Universal*. From a metaphysical point of view, this choice allows for the construction of a parsimonious ontology, based on the primitive and formally defined notion of *existential dependence*: We have that a particular x is *existentially dependent* (*ed*) on another particular y iff, as a matter of necessity, y must exist whenever x exists. Existential dependence is a modally constant relation, i.e., if x is dependent on y , this relation holds between these two specific particulars in all possible worlds in which x exists.

The word *Moment* is derived from the german *Momente* in the writings of E. Husserl and it denotes, in general terms, what is sometimes named *trope*, *abstract particular*, *individual accident*, or *property instance*. Thus, in the scope of this work, the term bears no relation to the notion of time instant in colloquial language.

Typical examples of moments are: a color, a connection, an electric charge, a social commitment. An important feature that characterizes all *moments* is that they can only exist in other particulars (in the way in which, for example, an electrical charge can exist only in some conductor). To put it more technically, we say that moments are *existentially dependent* on other particulars. Existential dependence can also be used to differentiate intrinsic and relational moments: *intrinsic moments* are dependent of one single particular (e.g., color, a headache, a temperature); *relators* depend on a plurality of particulars (e.g., an employment, a medical treatment, a marriage). A special type of existential dependence relation that holds between a moment x and the particular y of which x depends is the relation of *inherence* (i). Thus, for a particular x to be a moment of another particular y , the relation $i(x, y)$ must hold between the two. For example, inherence glues your smile to your face, or the charge in a specific conductor to the conductor itself. Here, we admit that moments can inhere in other moments. Examples include the individualized time extension, or the graveness of a particular symptom. The infinite regress in the inherence chain is prevented by the fact that there are individuals that cannot inhere in other individuals, namely, *objects*.

Objects are particulars that possess (direct) spatial-temporal qualities and that are founded on matter. Examples of objects include ordinary entities of everyday experience such as an individual person, a dog, a house, a hammer, a car, Alan Turing and The Rolling Stones but also the so-called *Fiat Objects* such as the North-Sea and its proper-parts, postal districts and a non-smoking area of a restaurant. In contrast with moments, objects do not inhere in anything and, as a consequence, they enjoy a higher degree of independence. To state this precisely we say that: an object x is *independent* of all other objects which are disjoint from x , i.e., that do not share a common part with x , where **independent** $(x, y) =_{\text{def}} \neg \text{ed}(x, y) \wedge \neg \text{ed}(y, x)$. This definition excludes the dependence between an object and its *essential* and *inseparable parts* (Guizzardi, 2005), and the obvious dependence between an object and its essential moments.

To complete the Aristotelian Square, we consider here the categories of *object universal* and *moment universal*. We use the term universal here in a broader sense without making any *a priori* commitment to a specific theory of universals. A universal thus can be considered here simply as something (i) which can be predicated of other entities and (ii) that can potentially be represented in language by *predicative terms*. We also use the relation $::$ of classification between particulars and universals. Object universals classify objects and moment universals classify moments. Examples of the former include Apple, Planet and Person. Examples of the latter include Color, Electric Charge and Headache. This distinction is also present in Aristotle's original differentiation between what is *said of a subject* (*de subjecto dici*), denoting classification and what is *exemplified in a subject* (*in subjecto est*), denoting inherence. Thus, the linguistic difference between the two meanings of the copula "is" reflects an ontological one. For example, the ontological interpretation of the sentence "Jane is a Woman" is that the Object Jane is classified by the Object kind Woman. However, when saying that "Jane is tall" or "Jane is laughing" we mean that Jane *exemplifies* the moment universal Tall or Laugh, by

virtue of her specific height or laugh. Finally, we define the relation of *characterization* between moment universals and the particulars that exemplify them: a moment universal M characterizes a universal U iff every instance of U exemplifies M. The categories of object, moment, object and moment universals as well as the relations of classification, inherence, exemplification and characterization are organized in terms of the so-called Aristotelian Square in Fig. 8.2.

8.2.2 Qualities, Qualia and Modes

An attempt to model the relation between intrinsic moments and their representation in human cognitive structures is presented in the theory of *conceptual spaces* introduced in Gärdenfors (2000). The theory is based on the notion of *quality dimension*. The idea is that for several perceivable or conceivable moment universals there is an associated quality dimension in human cognition. For example, height and mass are associated with one-dimensional structures with a zero point isomorphic to the half-line of nonnegative numbers. Other properties such as color and taste are represented by multi-dimensional structures.

In Gärdenfors (2000), the author distinguishes between *integral* and *separable* quality dimensions: “certain quality dimensions are integral in the sense that one cannot assign an object a value on one dimension without giving it a value on the other. For example, an object cannot be given a hue without giving it a brightness value (. . .) Dimensions that are not integral are said to be separable, as for example the size and hue dimensions.” He then defines a *quality domain* as “a set of integral dimensions that are separable from all other dimensions” and a *conceptual space* as a “collection of one or more domains” (Gärdenfors, 2000). Finally, he defends that the notion of conceptual space should be understood literally, i.e., quality domains are endowed with certain geometrical structures (topological or ordering structures) that constrain the relations between its constituting dimensions. In Gärdenfors (2000), the perception or conception of an intrinsic moment can be represented as a point in a quality domain. Following Masolo et al. (2003), this point is named here a *quale*.

An example of a quality domain is the set of integral dimensions related to color perception. A color quality *c* of an apple *a* takes its value in a three-dimensional color domain constituted of the dimensions hue, saturation and brightness. The geometric structure of this space (the color spindle (Gärdenfors, 2000)) constrains the relation between some of these dimensions. In particular, saturation and brightness are not totally independent, since the possible variation of saturation decreases as brightness approaches the extreme points of black and white, i.e., for almost black or almost white, there can be very little variation in saturation. A similar constraint could be postulated for the relation between saturation and hue. When saturation is very low, all hues become similarly approximate to grey.

We adopt in this work the term *quality structures* to refer to quality dimensions and quality domains, and we define the formal relation of *association* between a quality structure and a moment universal. Additionally, we use the terms *quality*

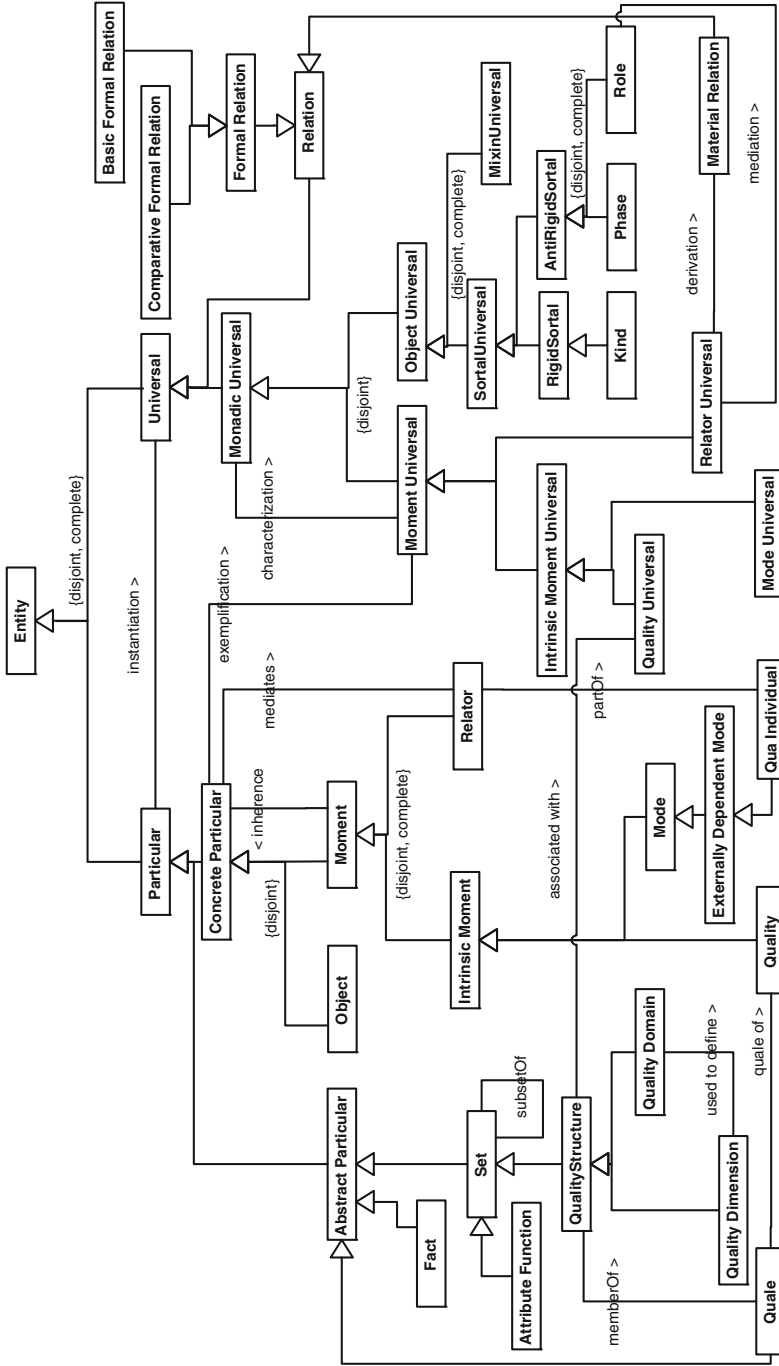


Fig. 8.2 A fragment of the unified foundational ontology (UFO)

universals for those intrinsic moment universals that are *associated with* a quality structure, and the term *quality* for a moment classified under a quality universal. We also assume that quality structures are always associated with a unique quality universal, i.e., a quality structure associated with the universal Weight cannot be associated with the universal Color.

Following Masolo et al. (2003), we take that whenever a quality universal Q is related to a quality domain D , then for every particular quality $x::Q$ there are *indirect qualities* inhering in x for every quality dimension associated with D . For instance, for every particular quality c instance of Color there are quality particulars h, s, b which are instances of quality kinds Hue, Saturation and Brightness, respectively, and that inhere in c . The qualities h, s, b are named *indirect qualities* of c 's bearer. Qualities such as h, s, b are named *simple qualities*, i.e., qualities which do not bear other qualities. In contrast, a quality such as c , is named a *complex quality*. Since the qualities of a complex quality $x::Q$ correspond to the quality dimensions of the quality domain associated with Q , then we have that no two distinct qualities inhering a complex quality can be of the same type. For the same reason, since there are not multidimensional quality dimensions, we have that complex qualities can only bear simple qualities. Moreover, we use the predicate *qualeOf*(x,y) to represent the formal relation between a quality particular y and its quale x .

Finally, we make a distinction between qualities and another sort of intrinsic moment named here *modes*. Modes are moments whose universals are not directly related to quality structures. In Gärdenfors (2000), the author makes the following distinction between what he calls *concepts* and *properties* (which at first could be thought to correspond to the distinction between Object and Moment universals, respectively): “Properties. . . form as special case of concepts. I define this distinction by saying that a *property* is based on *single domain*, while a *concept* may be based on *several domains*”. We claim, however, that only moment universals that are conceptualized w.r.t. a single domain, i.e., quality universals, correspond to properties in the sense of Gärdenfors (2000). There are, nonetheless, moment universals that as much as object universals can be conceptualized in terms of multiple separable quality dimensions. Examples include beliefs, desires, intentions, perceptions, symptoms, skills, among many others. Like objects, modes can bear other moments, and each of these moments can refer to separable quality dimensions. However, since they are moments, differently from objects, modes are necessarily existentially dependent of some particular.

8.2.3 Relations, Relators and Qua Individuals

Relations are entities that glue together other entities. In the philosophical literature, two broad categories of relations are typically considered, namely, *material* and *formal* relations (Heller and Herre, 2004; Smith and Mulligan, 1986). Formal relations hold between two or more entities directly, without any further intervening particular. In principle, the category of formal relations includes those relations

that form the mathematical superstructure of our framework including existential dependence (*ed*), inherence (*i*), *part-of* (<), *subset-of*, *instantiation*, *characterization*, *exemplification*, among many others not discussed here (Guizzardi, 2005). We name these relations here *basic formal relations* (Heller and Herre, 2004) or *internal relations* (Schneider, 2002). In this case, in conformance with Schneider (2002) we deem the tie (or nexus) between the relata as non-analyzable.

However, we also classify as formal those domain relations that exhibit similar characteristics, i.e., those relations of *comparison* such as *is taller than*, *is older than*, *knows more Greek than*. We name these relations *comparative formal relations*. As pointed out in Smith and Mulligan (1986), the entities that are immediate relata of such relations are not objects but intrinsic moments. For instance, the relation *heavier-than* between two atoms is a formal relation that holds directly as soon as the relata (atoms) are given. The truth-value of a predicate representing this relation depends solely on the atomic number (a quality) of each atom and the material content of *heavier-than* is as it were distributed between the two relata. Moreover, to quote Mulligan and Smith, “once the distribution has been effected, the two relata are seen to fall apart, in such a way that they no longer have anything specifically to do with each other but can serve equally as terms in a potentially infinite number of comparisons”.

Material relations, conversely, have material structure of their own and include examples such as *working at*, *being enrolled at*, and *being connected to*. Whilst a formal relation such as the one between Paul and his knowledge *x* of Greek holds directly and as soon as Paul and *x* exist, for a material relation of *being treated in* between Paul and the medical unit MU_1 to exist, another entity must exist which *mediates* Paul and MU_1 . We name these entities *relators*. Relators are particulars with the power of connecting entities. For example, a medical treatment connects a patient with a medical unit; an enrollment connects a student with an educational institution; a covalent bond connects two atoms. The notion of relator (relational moment) is supported by several works in the philosophical literature (Heller and Herre, 2004; Smith and Mulligan, 1986; Lowe, 2006) and, the position advocated here is that they play an important role in answering questions of the sort: what does it mean to say that John is married to Mary? Why is it true to say that Bill works for Company X but not for Company Y?

An important notion for the characterization of relators (and, hence, for the characterization of material relations) is the notion of *foundation*. Foundation can be seen as a type of *historical dependence* (Ferrario and Oltramari, 2004), in the way that, for example, an instance of *being kissed* is founded on an individual *kiss*, or an instance of *being punched by* is founded on an individual *punch*, an instance of *being connected to* between airports is founded on a particular flight connection. Suppose that John *is married to* Mary. In this case, we can assume that there is a particular relator (relational moment) m_1 of type *marriage* that mediates John and Mary. The foundation of this relator can be, for instance, a wedding event or the signing of a social contract between the involved parties. In other words, for instance, a certain event e_1 in which John and Mary participate can create a particular marriage m_1

which existentially depends on John and Mary and which mediates them. The event e_1 in this case is the foundation of relator m_1 and, m_1 is the so-called truthmaker of the propositions “John is married to Mary”.

Using this example, we can further elaborate on the nature of the relator m_1 . There are many moments that John acquires by virtue of being married to Mary. For example, imagine all the legal responsibilities that John has in the context of this relation. These newly acquired properties are intrinsic moments of John which, therefore, inhere and are existentially dependent on him. However, these moments also depend on the existence of Mary. We name this type of moment *externally dependent moment*, i.e., externally dependent moments are intrinsic moments that inhere in a single particular but that are existentially dependent on (possibly a multitude of) other particulars: a moment x is externally dependent iff it is existentially dependent of a particular which is *independent* (in the technical sense of 8.2.1) of its bearer.

In the case of a material externally dependent moment x there is always a particular *external* to its bearer (i.e., which is not one of its parts or intrinsic moments), which is the foundation of x . Again, in the given example, we can think of a certain event e_1 (wedding event or signing of social contract) in which both John and Mary participate and which founds the existence of these externally dependent moments inhering in John. Now, we can define a particular that bears all externally dependent moments of John that share the same external dependencies and the same foundation. We term this particular a *qua individual* (Masolo et al., 2005). Qua individuals are, thus, treated here as a special type of *complex externally dependent modes*. In this case, the complex mode inhering in John that bears all responsibilities that John acquires by virtue of a given wedding event can be named *John-qua-husband*.

To continue with the same example, we can think about another qua individual *Mary-qua-wife* which is a complex mode bearing all responsibilities that Mary acquires by virtue of the same foundation and that albeit inhering in Mary are also existentially dependent on John. The qua individuals *John-qua-husband* and *Mary-qua-wife* are existentially dependent on each other. Now, we can define an aggregate m_1 composed of these two qua individuals that share the same foundation, i.e., (*John-qua-husband* $<$ m_1) and (*Mary-qua-wife* $<$ m_1). In this example, m_1 is exactly the instance of the relational property *marriage* that mediates John and Mary and that makes true propositions such as “John is married to Mary”, “Mary is married to John”, “John is the husband of Mary”, and “Mary is the wife of John”.

In this example, a particular instance of the relational property marriage (i.e., a particular marriage relator) is the sum of all instantiated responsibilities that the involved parties acquire by virtue of a common foundation. In general, a relator can be defined as the aggregation of a number of qua individuals that share the same foundation. A relator is said to mediate (or connect) the relata of a material relation. Formally we have that: let x , y and z be three distinct individuals such that (a) x is a relator; (b) z is a qua individual and z is part of x ; (c) z inheres in y . In this case, we say that x *mediates* y , symbolized by $m(x, y)$. Additionally, we require that a relator mediates at least two distinct particulars. Again, using the example above, we say

that the particular relator marriage m_1 mediates the objects John and Mary and, for this reason, we can say that John and Mary are married to each other.

Analogous to the relation of characterization, we define a relation of *mediation* that can obtain between a set of object universal and a relator universal in the following way: If a relator universal U_R mediates the object universals $S_1 \dots S_n$, then every instance of U_R is existentially dependent on a plurality of entities, namely, particular instances of $S_1 \dots S_n$. Relator universals constitute the basis for defining material relations R. Material relations are themselves universals whose instances are n-tuples of particulars. We define the formal relation of derivation $derivation(R, U_R)$ holding between a relator universal U_R and a material relation R such that a n-tuple $\langle x_1 \dots x_n \rangle$ instantiates R iff there is a relator $r: U_R$ such that r mediates every x_i . To employ once more the example above, we have that $\langle \text{John}, \text{Mary} \rangle$ is an instance of both *married to* and *is the husband of*, and $\langle \text{Mary}, \text{John} \rangle$ is an instance of both *married to* and *is the wife of* because there is an individual marriage relator m_1 that mediates John and Mary.

8.2.4 Object Universals

Here we considered a fundamental distinction in the category of object Universals, namely, the one between *Sortal* and *Mixin Universals*. Whilst all universals carry a *principle of application*, only sortals carry a *principle of identity* for their instances. A principle of application is a principle for which we can judge whether a particular is an instance of that universal. In contrast, a principle of identity is a principle for which we can judge whether two particulars are the same. As an illustration of this point, contrast the two universals Apple and Red¹ instantiated by two particulars x and y: both universals supply a principle for which we can judge whether x and y are classified under those types (i.e., whether they are Apples, or Reds). However, only Apple supplies a principle for which we decide whether x and y are the same (i.e., merely knowing that x and y are both red gives no clue to decide whether or not $x = y$).

Within the category of sortal universals, we make a further distinction based on the formal notions of *rigidity* and *anti-rigidity*: A universal U is rigid if for every instance x of U, x is necessarily (in the modal sense) an instance of U. In other words, if x instantiates U in a given world w, then x must instantiate U in every possible world w'. In contrast, a universal U is anti-rigid if for every instance x of U, x is *possibly* (in the modal sense) not an instance of U. In other words, if x instantiates U in a given world w, then there must be a possible world w' in which x does not instantiate U. A sortal universal which is rigid is named here a *Kind*. In contrast, an anti-rigid sortal universal is termed here a *Phased-Sortal*. The prototypical example highlighting the modal distinction between these two categories is the

¹Red is used here as an object universal whose instances are particulars like a red apple x, not as a quality universal whose instances are particulars such as the specific redness of x (Guizzardi, 2005).

difference between the Kind Person and the Phase-Sortals Student and Adolescent instantiated by the particular John in a given circumstance. Whilst John can cease to be a Student and Adolescent (and there were circumstances in which John was not one), he cannot cease to be a Person. In other words, while the instantiation of the phased-sortals Student and Adolescent has no impact on the identity of a particular, if a particular ceases to instantiate the universal Person, then she ceases to exist as the same particular.

John can move in and out of the Student universal, while being the same particular, i.e. without losing his identity. This is because the principle of identity that applies to instances of Student and, in particular, that can be applied to John, is the one which is supplied by the kind Person of which the phased-sortals Student is a subtype. This is always the case with Phased-Sortals, i.e., for every phased-sortal PS, there is a unique ultimate kind K, such that: (i) PS is a specialization of K; (ii) K supplies the unique principle of identity obeyed by the instances of PS. If PS is a phased-sortal and K is the kind specialized by PS, there is a *specialization condition* φ such that x is an instance of PS iff x is an instance of K that satisfies φ . A further clarification on the different types of specialization conditions allows us to distinguish between two different types of phased-sortals: *Phases* and *Roles*. Phases constitute possible stages in the history of a particular. Examples include: (a) Alive and Deceased: as possible stages of a Person; (b) Catterpillar and Butterfly of a Lepidopteran; (c) Town and Metropolis of a City; (d) Boy, Male Teenager and Adult Male of a Male Person. Roles differ from phases with respect to the specialization condition φ . For a phase Ph, φ represents a condition that depends solely on intrinsic properties of Ph. For instance, one might say that if John is a Living Person then he is a Person who has the property of being alive or, if Spot is a Puppy then it is a Dog who has the property of being less than one year old. For a role Rl, conversely, φ depends on extrinsic (relational) properties of Rl. For example, one might say that if John is a Student then John is a Person who is enrolled in some educational institution, if Peter is a Customer then Peter is a Person who buys a Product x from a Supplier y , or if Mary is a Patient then she is a Person who is treated in a certain medical unit. In other words, an entity plays a role in a certain context, demarcated by its relation with other entities. This meta-property of Roles is named *Relational Dependence* and can be formally characterized as follows: A universal T is relationally dependent on another universal P via relation R iff for every instance x of T there is an instance y of P such that x and y are related via R. In other words, instances of T and P must be mediated by an instance of the relator universal U_R that induces the material relation R.

Finally, in Guizzardi (2005), we have formally proved a number of constraints involving these categories. These include (among a number of others): (i) a rigid universal cannot have as its superclass an anti-rigid one (consequently, a phased-sortal cannot subsume a kind in our theory); (ii) every object must instantiate exactly one kind (i.e., exactly one rigid independent sortal); (iii) a mixin cannot be subsumed by a sortal; (iv) a mixin cannot have direct instances.

The discussion of this section is summarized as follows: *Kinds* are *rigid, independent sortals* that supply a principle of identity for their instances; *Phases*

are *independent anti-rigid sortals*; *Roles* are *anti-rigid* and *relationally dependent sortals*, *Mixins* are *non-sortals*.

8.3 A Framework for Language Evaluation and (Re)Design

One of the main success factors behind the use of a modeling language lies in the language’s ability to provide to its target users a set of modeling primitives that can directly express relevant domain concepts, comprising what we name here a *domain conceptualization*. The elements constituting a *conceptualization* of a given domain are used to articulate abstractions of certain state of affairs in reality. We name the latter *domain abstractions*. Take as an example the domain of genealogical relations in reality. A certain conceptualization of this domain can be constructed by considering concepts such as *Person*, *Man*, *Woman*, *Father*, *Mother*, *Offspring*, *being the father of*, *being the mother of*, among others. By using these concepts, we can articulate a domain abstraction (i.e., a mental model) of certain facts in reality such as, for instance, that *a man named John is the father of another man named Paul*.

Conceptualizations and Abstractions are immaterial entities that only exist in the mind of the user or a community of users of a language. In order to be documented, communicated and analyzed they must be captured, i.e. represented in terms of some concrete artifact. This implies that a language is necessary for representing them in a concise, complete and unambiguous way. Figure 8.3 depicts the distinction between an abstraction and its representation, and their relationship with conceptualization and representation language. In the scope of this work the representation of a domain abstraction in terms of a representation language \mathcal{L} is called a *model* and the language \mathcal{L} used for its creation is called a *modeling language*.

In order for a model \mathcal{M} to faithfully represent an abstraction \mathcal{A} , the modeling primitives of the language \mathcal{L} used to produce \mathcal{M} should faithfully represent the domain conceptualization \mathcal{C} used to articulate the represented abstraction \mathcal{A} . The *Domain Appropriateness* of a language is a measure of the suitability of a

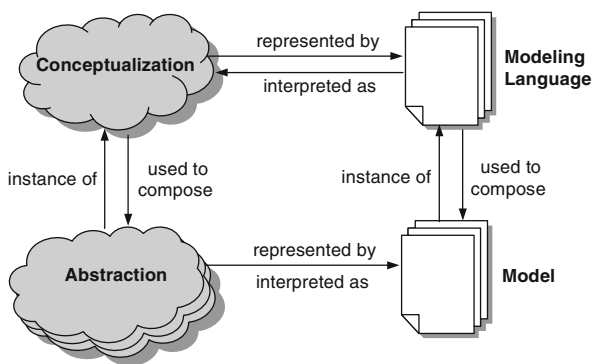


Fig. 8.3 Relations between conceptualization, abstraction, modeling language and model

language to model phenomena in a given domain, or in other words, of its truthfulness to a given domain in reality. On a different aspect, different languages and specifications have different measures of pragmatic adequacy (Guizzardi, 2005). *Comprehensibility appropriateness* refers to how easy is for a user of a given language to recognize what that language's constructs mean in terms of domain concepts and, how easy is to understand, communicate and reason with the specifications produced in that language. These two quality criteria can be systematically evaluated by comparing, on one hand, a concrete representation of the worldview underlying that language (captured by that language's *metamodel*) to, on the other hand, a concrete representation of a domain conceptualization, or a *domain ontology*. The truthfulness to reality (*domain appropriateness*) and conceptual clarity (*comprehensibility appropriateness*) of a modeling language depend on the level of homomorphism between these two entities (Guizzardi, 2005). The stronger the match between an abstraction in reality and its representing model, the easier is to communicate and reason with that model.

The mapping from concepts-to-constructs and its inverse (i.e., constructs-to-concept) are named here a *representation* and *interpretation* mappings, respectively. In Guizzardi (2005), we discuss a number of properties that should be reinforced for isomorphic mappings to take place between an ontology \mathcal{O} representing a domain \mathcal{D} and a language's metamodel. If isomorphism can be guaranteed, the implication for the human agent who interprets a diagram (model) is that his interpretation correlates precisely and uniquely with an abstraction being represented. By contrast, where the correlation is not an isomorphism then there may potentially be a number of unintended abstractions which would match the interpretation. These properties are briefly discussed in the sequel and are illustrated in Fig. 8.4: (a) *Soundness*: A language \mathcal{L} is *sound* w.r.t. to a domain \mathcal{D} iff every modeling primitive in the language has an interpretation in terms of a domain concept in the ontology \mathcal{O} ; (b) *Completeness*: A language \mathcal{L} is *complete* w.r.t. to a domain \mathcal{D} iff every

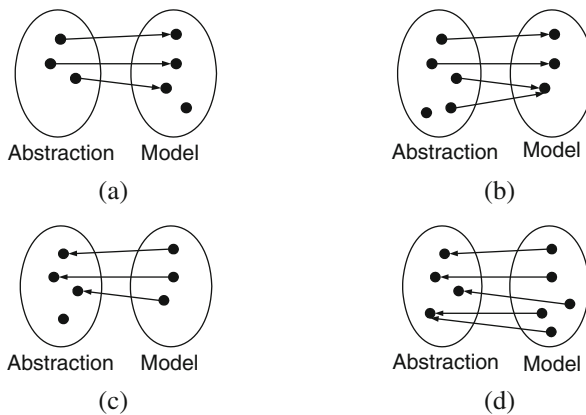


Fig. 8.4 Examples of lucid (a) and sound (b) representational mappings from abstraction to model; examples of laconic (c) and complete (d) interpretation mappings from model to abstraction

concept in the ontology \mathcal{O} of that domain is represented in a modeling primitive of that language; (c) *Lucidity*: A language \mathcal{L} is *lucid* w.r.t. to a domain \mathcal{D} iff every modeling primitive in the language represents at most one domain concept in \mathcal{O} . (d) *Laconicity*: A language \mathcal{L} is *laconic* w.r.t. to a domain \mathcal{D} iff every concept in the ontology \mathcal{O} of that domain is represented at most once in the metamodel of that language. In Guizzardi (2005), we also provide a methodological framework for systematically assessing these properties given a language and a domain.

Unsoundness, Non-Lucidity, Non-Laconicity and *Incompleteness* violate what the philosopher of language Grice (1975) names *conversational maxims* that states that a speaker is assumed to make contributions in a dialogue which are *relevant, clear, unambiguous, and brief, not overly informative and true according to the speaker's knowledge*. Whenever models do not adhere to these conversational maxims, they can communicate incorrect information and induce the user to make incorrect inferences about the semantics of the domain.

In regards to the property of completeness, when mapping the elements of a domain ontology to a language metamodel we must guarantee that these elements are represented in their full formal descriptions. In other words, the metamodel \mathcal{MT} of language \mathcal{L} representing the domain ontology \mathcal{O} must also represent this ontology's full axiomatization. In formal, model-theoretic terms, this means that these entities should have the same set of logical models. In Guizzardi (2005), we discuss this topic in depth and present a formal treatment of the idea. The set of logical models of \mathcal{O} represent the state of affairs in reality deemed possible by a given domain conceptualization. In contrast, the set of logical models of \mathcal{MT} stand for the world structures which can be represented by the grammatically correct specifications of language \mathcal{L} . In summary, we can state that if a domain ontology \mathcal{O} is fully represented in a language metamodel \mathcal{MT} of \mathcal{L} , then the only grammatically correct models of \mathcal{L} are those which represent state of affairs in reality deemed possible by the domain conceptualization represented by \mathcal{O} (termed *intended world structures*).

In the beginning of this section, we have exemplified the notions discussed above by referring to the domain of genealogical relations. This exemplifies what is named a *material domain* in the literature. Accordingly, a modeling language designed to represent phenomena in this domain is named a *Domain-Specific Modeling Language*. However, take the case of a (domain-independent) *general conceptual modeling language* (e.g., EER, ORM, UML). What should be real-world conceptualization that this language should commit to? The position defended here is that it should be a system of general categories and their ties, which can be used to articulate domain-specific common sense theories of reality. This meta-conceptualization should comprise a number of domain-independent theories (e.g., types and instantiation, taxonomic structures, identity, existential dependence, etc.) which are able to characterize aspects of real-world entities irrespective of their particular nature. The development of such general theories of reality is the business of the philosophical discipline of *Formal Ontology* in philosophy and a concrete artifact representing one of these meta-conceptualizations is a *Foundational Ontology*. An example of a Foundational Ontology is the UFO Ontology presented in the Section 8.2.

8.4 Evaluating and Redesigning the UML 2.0 Metamodel

In the sequel we start by constructing representation and interpretation mappings between the concrete metaclasses of the UML metamodel presented in Object Management Group (2003) and the ontological categories comprising the foundational ontology depicted in Fig. 8.2.

Class and Generalization: We begin by focusing on a special sense of the UML metaclass *Class* (see Fig. 8.5). By class hereby we mean the notion of a first-order class, as opposed to *powertypes*, and one whose instances are single objects, as opposed to *association classes*, whose instances are tuples of objects. In this sense, if we make a representation mapping from UFO to the UML metamodel, we can map the category of *Monadic Universal* to the UML element of a *Class*. However, by carrying on this process, we realize that in UML there are no modeling constructs that represent the ontological categories specializing *Object Universal* in Fig. 8.2. In other words, there are ontological concepts prescribed by our reference ontology that are not represented by any modeling construct in the language. This amounts to a case of *incompleteness*. Moreover, as discussed in Section 8.2.4, the theory of object universals comprising UFO prescribes a number of constraints governing the relations between these different types of universals. By not taking this into account,

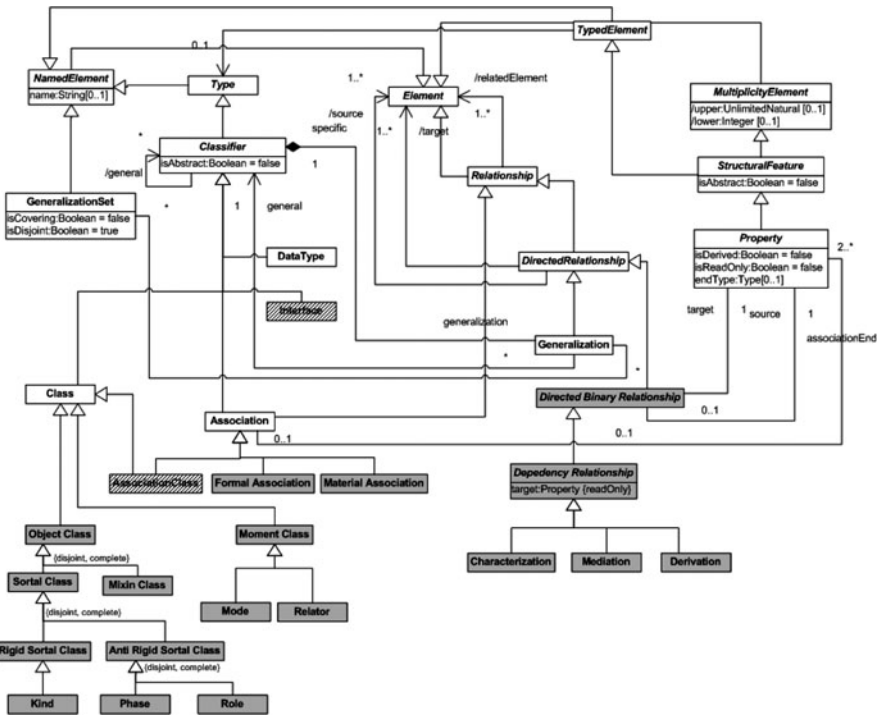


Fig. 8.5 The redesigned UML 2.0 metamodel

the UML metamodel admits a number of grammatically correct specifications and logical models which are not representations of valid state of affairs (intended world structures) according to the reference ontology.

Attributes, Attribute Values and Datatypes: If we now consider the other types of monadic universals accounted in our theory, namely, *moment* and *relator universals* we shall realize that they too lack representation in the language metamodel. This amounts to another case of *incompleteness* in the modeling language.

As discussed at length in Guizzardi (2005), quality universals are typically not represented in a conceptual model explicitly but via *attribute functions* that map each of their instances to points in a quality structure. For example, suppose we have the universal Apple (an object universal) whose instances exemplify the universal Weight. We say in this case that the quality universal Weight *characterizes* the kind Apple. Thus, for an arbitrary instance x of Apple there is a quality w (instance of the quality universal Weight) that inheres in x . Associated with the universal Weight, and in the context of a given measurement system (e.g., the human perceptual system), there is a quality dimension *weightValue*, which is a set isomorphic to the half line of positive integers obeying the same ordering structure. Quality structures are taken here to be theoretical abstract entities modeled as sets. In this case, we can define an *attribute function* (another abstract theoretical entity) *weight* (Kg), which maps each instance of apple (and in particular x) onto a point in a quality dimension, i.e., its *quale*. Thus, attribute functions are the ontological interpretation of UML attributes, i.e., UML Properties which are owned by a given classifier (Fig. 8.5).

As any property, a UML attribute is a typed element and, thus, it is associated to Type. Type constrains the sort of entities that can be assigned to slots representing that attribute in instances of their owning classifier. Since Classifier is a specialization of Type, we have that both Classes and Datatypes can be the associated types of an UML attribute. In other words, an attribute represents both an attribute function and a sort of a *relational image function*² that, for example, in the binary relation *ownership* between the classifiers Person and Car, maps a particular Person p to all instances of Car that are associated with p via this relation (i.e., all cars owned by p). From a software design and implementation point of view, an attribute represents a method implemented by the owning class, and the type of the attribute represents the returning type of that method. However, from a conceptual point of view, in the UML metamodel an attribute stands both for a monadic (intrinsic) and for a relational property and, thus, it can be considered a case of *non-lucidity*. On another perspective, UML offers an alternative notation for the representation of attributes, namely, *navigable end names*. That is, the same ontological concept (*attribute function*) is represented in the language via more than one construct, which characterizes a case of *non-laconicity*.

²A relational image function is formally defined as follows: Let R be a binary relation defined for the two sets X and Y . The function \mathbf{Im} with signature $\mathbf{Im}(_, _): X \times (X \Leftrightarrow Y) \rightarrow \wp(Y)$ is defined as $\mathbf{Im}(x, R) = \{y | (x, y) \in R\}$.

The `DataType` associated with an attribute `A` of class `C` is the representation of the quality structure that is the co-domain of the attribute function represented by `A`. In other words, a quality structure is the ontological interpretation of the UML `DataType` construct. Moreover, we have that a multidimensional quality structure (quality domain) is the ontological interpretation of the so-called *Structured DataTypes*. Quality domains are composed of multiple integral dimensions. This means that the value of one dimension cannot be represented without representing the values of others. The fields of a datatype representing a quality domain `QD` represent each of its integral quality dimensions. Alternatively we can say that each field of a datatype should always be interpreted as representing one of the integral dimensions of the `QD` represented by the datatype. The constructor method of the `dataType` representing a quality domain must reinforce that its tuples always have values for all the integral dimensions. Finally, an algebra can be defined for a `DataType` so that the relations constraining and informing the geometry of represented quality dimensions are also suitably characterized. As discussed in Guizzardi (2005), according to the UML specification, a `DataType` is an abstract entity that collects other abstract entities (“*pure values*”) that can be multiply referred, i.e., a `DataType` is not a multiply instantiated universal but an abstract particular (set) with other particulars as members.

Associations: In UML, the association meta-construct is used to represent the ontological concept of *Relation*. Relations in UFO can be *material* or *formal*. The latter in turn can be subdivided in *basic formal relations* (internal relations) and *comparative formal relations*. Since class diagrams only represent universals, the only basic formal relations among the ones we have considered that should have a representation in these models are the relations of *characterization*, *mediation* and *derivation*. These concepts have no representation in the UML metamodel, which characterizes another case of *incompleteness*.

The association class construct in UML exemplifies a case of *non-lucidity*, since “an association class can have as instances either (a) a n -tuple of entities which classifiers are endpoints of the association; (b) a $n+1$ -tuple containing the entities which classifiers are endpoints of the association plus an instance of the objectified association itself” (Breu et al., 1997). This is to say that an association class can be interpreted both as a *relation* and what is termed in the literature a *factual universal* (Guizzardi, 2005). In short, if the relator r connects (mediates) the entities a_1, \dots, a_n then this yields a new particular that is denoted by $\langle r: a_1, \dots, a_n \rangle$. Particulars of this latter sort are called *material facts*.

In addition to that, since the “*instance of the objectified association itself*” is supposed to be an object identifier for the n -tuple, one cannot represent cases in which the same relator mediates multiple occurrences of the same n -tuple. As an example of the latter suppose the following situation. Suppose a `Treatment` relator universal and a `TreatedIn` material relation (derived from it) defined between `Patients` and `Medical Units`. Now suppose that treatment t_1 mediates the individuals John, and the medical units `MedUnit#1` and `MedUnit#2`. In this case, we have as instances of `Treatment` both facts $\langle t_1: \text{John}, \text{MedUnit}_{\#1} \rangle$ and $\langle t_1: \text{John}, \text{MedUnit}_{\#2} \rangle$. However, this cannot be represented in such a manner in UML. In UML, t_1 is supposed

to function as an object identifier for a unique tuple. Thus, if the fact $\langle t_1: \text{John}, \text{MedUnit}_{\#1} \rangle$ holds then $\langle t_1: \text{John}, \text{MedUnit}_{\#2} \rangle$ does not, or alternatively, John and $\text{MedUnit}_{\#2}$ must be mediated by another relator. These are, nonetheless, unsatisfactory solutions, since it is the very same relator Treatment that connects one patient to a number of different medical units. In conclusion, association classes on one hand represent a case of non-lucidity, on the other hand, allow for a case of construct incompleteness at the instance level.

Interfaces: According to the UML specification, an interface is a declaration of a coherent set of features and obligations. It can be seen as a kind of contract that partition and characterize groups of properties which must be fulfilled by any instance of a classifier that implements that interface. In an interpretation mapping from the UML metamodel to UFO, an interface qualifies as a case of *unsoundness*. This means that, being merely a design and implementation construct, there is no category in the conceptual modeling ontology proposed here that serve as the ontological interpretation for a UML interface.

8.5 Reinforcing the Isomorphism Between UFO and UML

As demonstrated in the previous section, from an ontological point of view, UML includes cases of ontological incompleteness, unsoundness, non-lucidity and non-laconicity. In the sequel, we discuss briefly how these problems have been solved to produce an ontologically well-founded version of UML for conceptual modeling (Guizzardi, 2005).

Incompleteness: In order to remedy this problem, we propose extensions to the UML metaclass *Class* that represent different types of monadic universals. As shown in Fig. 8.5, these extensions represent finer-grained distinctions between different sorts of *object types* as well as the notions of *mode* and *relator* universals and *material* and *formal* relations.


Another example of incompleteness identified in the previous section is w.r.t. the representation of different types of basic formal relations, namely, the relations of *characterization*, *mediation* and *derivation*. There are a number of common characteristics shared by these relations. Firstly, they are all *directed relations*. In the case of characterization, the source is a class representing a mode universal; in the case of mediation, one representing a relator universal; in the case of derivation, a material relation. In the first two cases, the target is a class representing either an object or moment universal, while in the case of derivation, the target is necessarily one representing a relator universal. Secondly, all these relations are mapped in the instance level to an *existential dependency* relation between the corresponding source particulars and their depended particulars. This has the following consequences in the metamodel: (i) the association end connected to the target class must have the cardinality constraints of one and exactly one, since every moment or fact is a dependent entity; (ii) the association end connected to the target class must have the meta-attribute *isreadOnly* = *true*, since existential dependency is modally constant; (iii) existential dependency relations are always binary relations.

In order to account to all these requirements, we extend the original UML meta-model by extending the metaclass direct relationship with the metaclasses direct binary relationship, dependency relationship, and finally, the basic formal relations of characterization, mediation and derivation (Fig. 8.5). Finally, since a relator is dependent (mediates) on at least two numerically distinct entities, we have the following additional constraint: (iv) Let R be a class representing a relator universal and let $\{C_1 \dots C_2\}$ be a set of classes mediated by R (related to R via a *mediation* relation). Finally, let $lower_{C_i}$ be the value of the minimum cardinality constraint of the association end connected to C_i in the mediation relation. Then, $\left(\sum_{i=1}^n lower_{C_i}\right) \geq 2$.

Asides from incorporating metaclasses that represent the missing ontological concepts, the extended UML metamodel must also include a number of constraints derived from the constraints in the ontology that restrict the ways the introduced elements can be related (see constraints on Section 8.2.4 as well as (i–iv) above). The goal is to have a metamodel such that all grammatically correct specifications according to this metamodel have logical models that are *intended world structures* of the conceptualizations they are supposed to represent. In Guizzardi (2005), besides from extending the UML meta-model in order to represent the ontological concepts discussed above, we define a *profile* that implements the metaclasses of this (extended) UML metamodel, as well as their interrelationships and constraints. By using this profile, for example, the concrete *object classes* in Fig. 8.5 are represented in conceptual models as stereotyped classes representing each of the considered ontological distinctions. Likewise, the admissible relations between these ontological categories, derived from the postulates of our theory, are represented in the profile as syntactical constraints governing the admissible relations between the corresponding stereotyped classes. A fragment of this profile is shown in Table 8.1. For the complete definition of this profile as well as an in depth discussion motivating its elements one should refer to Guizzardi (2005).

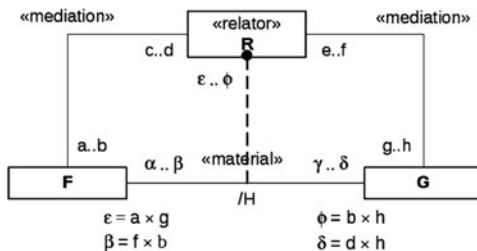
Non-Lucidity: As discussed in the previous section, in UML, attributes represent both the ontological concepts of attribute functions and relational image functions, which is case of non-lucidity. To eliminate this problem, we prescribe that attributes should only be used to represent attribute functions. As consequence, their associated types should be restricted to DataTypes only. The UML construct of association classes amounts to a case of both non-lucidity (since it represents a factual and relator universal) and incompleteness (since one cannot represent cases in which the same relator mediates multiple occurrences of the same n-tuple). We propose, therefore, to disallow the use of association classes in UML for the purpose of conceptual modeling. In contrast, we propose to represent relational properties explicitly. We use the stereotype «relator» to represent the ontological category of relator universals. Relator universals can induce material relations. A material relation induced by a relator universal R is represented by a UML association stereotyped as «material» (UML base class *association*). The basic formal relation *derivation* is represented by a dashed line with a black circle in one of the ends (see Fig. 8.6). A derivation relation is a specialized type of relationship between the stereotyped association representing the derived «material» association and the stereotyped class

Table 8.1 Fragment of the UML profile implementing the metamodel of Fig. 8.5

Metaclass		Description
«kind» A	Kind represents rigid, relationally independent object universals that supply a principle of identity for their instances.	
Constraints		
1. Every object represented in a conceptual model using this profile must be an instance of a kind, directly or indirectly. This means that every concrete element of this profile used in a class diagram (isAbstract = false) must include in its general collection one class stereotyped as «kind»; 2. An object represented in a conceptual model using this profile cannot be an instance of more than one kind. This means that any stereotyped class in this profile used in a class diagram must not include in its general collection more than one kind. Moreover, a kind must also not include another kind in its general collection; 3. A Class representing a rigid object universal cannot be a subclass of a Class representing an anti-rigid universal. Thus, a kind cannot have as a supertype (must not include in its general collection) a member of {«phase», «role»}.		
Metaclass		Description
Derivation Relation	Dependency Relationship	A derivation relation represents the formal relation of derivation that takes place between a material relation and the relator universal this material relation is derived from.
		
Constraints		
1. A derivation relation must have one of its association ends connected to a relator universal (the black circle end) and the other one connected to a material relation (self.target.oclIsTypeOf(Relator)=true, self.source.oclIsTypeOf(Material Association)=true); 2. derivation associations are always binary associations; 3. The black circle end of the derivation relation must have the cardinality constraints of one and exactly one (self.target.lower = 1 and self.target.upper = 1); 4. The black circle end of the derivation relation must have the property (self.target.isReadOnly = true); 5. The cardinality constraints of the association end connected to the material relation in a derivation relation are a product of the cardinality constraints of the «mediation» relations of the relator universal that this material relation derives from. This is done in the manner previously shown in this section. However, since «mediation» relations require a minimum cardinality of one on both of its association ends, then the minimum cardinality on the material relation end of a derivation relation must also be ≥ 1 (self.source.lower ≥ 1).		

representing the founding «relator» universal. The black circle represents the role of foundation of the relator universal side. Every «material» association must be the association end of exactly one derivation relation. Still on Fig. 8.6, from the cardinality constraints of the two «mediation» relations we can derive the maximum cardinality of the derivation relation (on the material relation end) and the cardinality constraints on both association ends of the material relation itself. For instance, the upper constraint δ on the end connected to G in the H relation is the result of $(d \times h)$; the upper constraint β in the end connected to F is the result of $(f \times b)$. The upper constraint ϕ in the end H of the derivation relation is the result of $(b \times h)$. We should highlight that the relator particular is the actual instantiation of the corresponding relational property (the objectified relation). Material relations stand merely for the facts derived from the relator particular and its mediating entities. Therefore, we claim that the representation of the relators of material relations must have primacy over the representation of the material relations themselves. In other words, the representation of

Fig. 8.6 Representing Material Relations and their founding relators



$\llcorner\text{material}\llcorner$ relations can be omitted but whenever a $\llcorner\text{material}\llcorner$ is represented it must be connected to an association end of a derivation relation.

Finally, we use the stereotype $\llcorner\text{formal}\llcorner$ to represent comparative formal relations. Comparative formal relations and material relations are derived relations. Whilst the former are derived from intrinsic properties of the related entities, the latter are derived from relators and their mediating entities. Therefore, we prescribe that UML associations stereotyped as $\llcorner\text{material}\llcorner$ must have the meta-attribute ($\text{isDerived} = \text{true}$). *Mutatis Mutandis*, we use the same meta-attribute to represent formal relations which are not *internal relations*, i.e., which are comparative.

Non-Laconicity: In the UML notation, the same ontological concept of attribute functions has two representations in terms of the language constructs, namely, the textual notation for attributes and navigable association ends. This situation could be justified from a pragmatic point of view if navigable ends were used to model only *structured DataTypes*, and if the textual notation for attributes were only used to model the simple ones. However, in the current UML metamodel, there is no constraint on using both notations for both purposes. To eliminate the potential ambiguity of this situation, we propose to use navigable ends to represent only attribute functions whose co-domains are *multidimensional quality structures* (quality domains). Conversely, those functions whose co-domains are quality dimensions should only be represented by the attributes textual notation.

Unsoundness: An example of a UML construct which lacks an ontological interpretation is the construct of *Interfaces*. For this reason, we propose that the use of this construct should be disallowed in an ontologically well-founded version of UML. In Fig. 8.5, the metaclasses interface and association classes which have been disallowed in this metamodel according to our analysis appear as hachured classes.

8.6 Final Considerations

The development of a well-grounded, axiomatized upper level ontology is an important step towards the definition of real-world semantics for conceptual modeling diagrammatic languages. In this paper, we use present the ontology *UFO* (*Unified*

Foundational Ontology), which has been designed with the specific purpose of serving as a foundational theory for conceptual modeling. Additionally, we briefly present an ontology-based framework for evaluating the *domain* and *comprehensibility appropriateness* of modeling languages. The framework defines a systematic method for comparing the *metamodel* of a language with a concrete representation of a conceptualization of a given subject domain, termed a *reference ontology*. The paper illustrates the usefulness of the UFO ontology as a reference ontology and application of the method by evaluating and redesigning the UML metamodel for the purpose of conceptual modeling.

In Guizzardi (2005), the redesigned UML metamodel discussed here has been used in the implementation of a UML profile for Conceptual Modeling. The profile comprises of: (i) a set of stereotypes representing ontological distinctions proposed by the theory (ii) constraints on the possible relations to be established between these elements, representing the postulates of the theory. By using this profile, we were able to propose a number of sound engineering tools and principles, and methodological guidelines for the practice of conceptual modeling such as the *role modeling design pattern* and the *visual patterns for delimiting the scope of transitive parthood relations*, both presented in Guizzardi (2005).

Finally, it is important to emphasize that, in this article, only a fragment of UFO is presented. In particular, a fragment of the Ontology of Endurants in UFO named UFO-A. In Guizzardi and Wagner (2005) and Guizzardi et al. (2008), UFO is presented in three compliance sets, namely, UFO-A: an *Ontology of Endurants*; UFO-B: an *Ontology of Perdurants*, and UFO-C, which is built upon UFO-A and B to compose an *Ontology of Social Concepts*. Although UFO-B and C do not enjoy the same level of maturity and stability as UFO-A, they have been recently employed with success in the analysis of other conceptual modeling languages and frameworks such as REA (Resource-Event Action) (Guizzardi and Wagner, 2005), Tropos and AORML (Guizzardi and Guizzardi, 2011), and the ODE Software Process Ontology (Guizzardi et al., 2008), among others.

References

- Breu, R., U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thurner. 1997. Towards a formalization of the unified modeling language. In Proceedings of the 11th ECOOP, Jyväskylä, Finland.
- Ferrario, R., and A. Oltramari. 2004. Towards a computational ontology of the mind. In Proceedings of the 3rd International Conference on Formal Ontology in Information Systems (FOIS), Torino, Italy.
- Gärdenfors, P. 2000. *Conceptual spaces: The geometry of thought*. Cambridge, MA: MIT Press.
- Grice, H.P. 1975. Logic and conversation. In *Syntax and semantics: vol 3. Speech acts*, eds P. Cole and J. Morgan, 43–58. New York, NY: Academic Press.
- Guizzardi, G. 2005. Ontological foundations for structural conceptual models, PhD Thesis, University of Twente, The Netherlands.
- Guizzardi, R.S.S., and G. Guizzardi. 2011. Ontology-based transformation framework from TROPOS to AORML. In *Social modeling for requirements engineering*, Cooperative Information Systems Series, eds P. Giorgini, N. Maiden, J. Mylopoulos, and E. Yu, Cambridge, MA: MIT Press, 547–570.

- Guizzardi, G., R. Falbo, and R.S.S. Guizzardi. 2008. Grounding software domain ontologies in the unified foundational ontology (UFO): The case of the ODE software process ontology. In Proceedings of the 11th Iberoamerican Conference on Software Engineering, Recife, Brazil.
- Guizzardi, G., and Wagner, G. 2005. On a unified foundational ontology and some applications of it in business modeling. In *Ontologies and business systems analysis*, eds. M. Rosemann, and P. Green. Hershey, PA: IDEA Publisher.
- Heller, B., and H. Herre. 2004. Ontological categories in GOL. *Axiomathes* 14:71–90.
- Lowe, E.J. 2006. *The four category ontology*. Oxford: Oxford University Press.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. Ontology library, WonderWeb Deliverable D18.
- Masolo, C., G. Guizzardi, L. Vieu, E. Bottazzi, and R. Ferrario. 2005. Relational roles and qua individuals. In AAI Fall Symposium on Roles, an Interdisciplinary Perspective, Virginia.
- Object Management Group. 2003. UML 2.0 superstructure specification, Doc.# ptc/03-08-02, Aug 2003.
- Schneider, L. 2002. Formalised elementary formal ontology, ISIB-CNR Technical Report 03/2002. <http://www.loa-cnr.it/Publications.html>
- Smith, B., and K. Mulligan. 1986. A relational theory of the act. *Topoi* 5(2):115–130.

Chapter 9

Lightweight Ontologies

John Davies

9.1 Introduction

Ontology was, in its original sense, a branch of philosophy and in this sense of the word is the study of what (kinds of) things exist: it seeks to identify (or posit) the categories of existence and the relationships between those categories; and to define or describe entities¹ within this framework. As such, the ultimate goal of ontology is to provide a definitive and complete classification of entities and the relations between those entities in all spheres of reality, both material and abstract. In this context, the phrase “lightweight ontology” has little meaning: indeed, the purpose of the enterprise is to be as “heavyweight” as to define a framework which is as complete and definitive a representation of reality as possible.

The term ontology has also, however, come to prominence over recent years in computer science and in particular in the areas of knowledge representation (KR) and reasoning and semantic web technology.

As described in more detail in Smith and Welty (2001), interest in ontology arose in 3 separate but related areas of computer science. At a high level, this interest arises from the realisation that the behaviour of a software system is meaningful only by virtue of the interpretation put on it by users of the system. As such, a system which employs a more accurate model of the user’s world will in general be more meaningful to that user and will be better able to accommodate change.

In the area of database management systems, it was found that, as database technology itself matured and stabilised, the issue of conceptual modelling was a more important and subtle one: the quality of a requirements specification and ultimately that of the resulting information system itself turned out to be heavily dependent on the ability of a developer to extract and represent accurately knowledge about the modelled domain.

J. Davies (✉)
BT Innovate, British Telecommunications plc, London, UK
e-mail: john.nj.davies@bt.com

¹We define an entity to be that which has a distinct separate existence.

Similarly, in software engineering the 1980s saw the emergence of object-oriented technologies which encouraged an approach to software development requiring the modelling of the application domain using (classes of) objects which could have associated with them both data and “methods” – software procedures which in some sense model an object’s capabilities. Objects can invoke the methods of other objects via a message-passing protocol. Programs are then seen as the behaviour of a set of cooperating objects rather than the traditional set of instructions to the computer. In such a paradigm, the issue of a formal and consistent basis for “object modelling” quickly arose.

In artificial intelligence (AI), the attempt to explicitly represent and reason about “knowledge” led to recognition of the relevance of work done in ontology. AI practitioners created knowledge bases representing ground facts and axioms about some domain, typically accompanied by a more or less formal mechanism for automated reasoning, allowing the derivation of new information in a specific instance. McCarthy (1980) stated that creators of intelligent systems must “list everything that exists, building an ontology of our world.”

Thus the KR community and, latterly, the semantic technology community began to use the term “ontology” in a somewhat different sense to its original meaning. An ontology increasingly came to mean a model of a domain of interest described in a logical language. Of course, for a software system, that which “exists” is precisely that which can be represented and processed (reasoned about) and so this focus on *logic-based* descriptions is to some extent natural. Logics provide a proof theory (a set of axioms and inference rules which can be used to make deductions) and a model theory (a formal analysis of the relationship between statements in the logic and the world being represented). Indeed, it can be argued that knowledge representation languages which do not have semantic models of the type furnished by logic-based approaches do not actually represent anything: without a clear account of what statements in the language claim to be true in the world, how are we to know what is being represented? Non-logical presentations only find their meaning when processed by computer programs: and different programs may process the same data differently because the lack of an explicit semantic model can lead to different (implicit) semantics being ascribed by different programs.

In addition to a clear semantic account, the use of logic offers other advantages including a clear understanding of the consistency and decidability of the logic at hand. Logics can be shown to be consistent: that is, providing the information represented in the logic initially is itself consistent, no inconsistency can arise from the application of the logical inference rules. Decidability is particularly important in the computational context, concerning as it does the tractability of theorems provers. A decidable logic is one wherein a theorem prover if given a statement to prove will be able to determine in a finite time whether or not the statement is true or false. Other interesting properties of logics are soundness and completeness which inform us about the relationship between the proof theory and model theory of a logic. If a logic is sound, all formulae which can be derived from a set of formulae F using the proof theory are true in all models in which F are true. If a logic is complete, everything true in a given model is provable by the corresponding proof theory.

The availability of these kinds of mathematical results about logics are invaluable in telling us the computational properties of systems we may implement based on them. Although these results are sometimes negative (as when a logic is incomplete or undecidable), the key point is that the results are known at all: for more ad-hoc representation schemes no such results are typically known. McDermott (1978) argues convincingly that these properties are important from more than a purely theoretical point of view. In essence, his point is that it is crucial that a system not only “works” but is *understood*.²

However, if we move away from the requirement that ontologies be machine-processable the picture is a little different. In the case where an ontology is used in order to improve the communication between different departments in a company, for example, a formal approach may not be necessary. Indeed, such an approach may be actually unhelpful: the majority of users of such an ontology are likely to have a greater shared understanding of an ontology expressed in a natural language than one expressed in a description logic, for example. Uschold (Building Ontologies: Towards a Unified Methodology, 1996) makes the distinction between the goal of an ontology and the need for formality. He suggests that when it is used to improve the communication between people, natural language can be sufficient. Our focus in this chapter, however, will be on *machine-processable* ontologies and, as argued above, therefore on *logic-based* approaches.

Smith (2003) discusses the relationship between ontology in its older philosophical sense and its later usage in computer science in more detail. For the purposes of this chapter, we focus on the meaning of the term as used by computer scientists, noting that, as mentioned above, this is the context in which the term “lightweight ontology” is meaningful.

The most commonly quoted definition of this sense of the term is that given by Gruber (1992, 1993):

An ontology is an explicit specification of a conceptualisation

In this sense, the word refers to a *software artefact*: a computer-processable model of some domain of interest. *Lightweight ontologies* in this context are such providing the simplest formalization of the simplest model, adequate for the task at hand. The rationale is that simple ontologies are often more appropriate and economical; they are easier to understand, adapt, management, update, and use. Lightweight ontologies can “survive” in computationally extreme environments where scale and performance are critical, e.g. very large databases and search engines.

In the next section, we consider this notion of ontology in a little more detail, before proceeding to distinguish lightweight ontologies from other ontologies. We

²Note that while advocating a logic-based approach we are not advocating any specific logic (such as description logic, for example). Indeed, currently available logics lack some of the properties which would seem to be required for representing and reasoning at the scale and lack of precision found on the web. [Fensel and van Harmelen 2007] discuss new, more appropriate inference mechanisms.

then motivate and describe techniques which have been developed to translate other schemes for information modelling into lightweight ontologies. In Section 9.3, we then consider the Semantic Web and its ontological languages in more detail before proceeding over the next 3 sections to discuss the application of ontologies to information integration, knowledge management and service-oriented environments. Section 9.7 contains some brief concluding remarks.

9.2 Lightweight Ontologies

9.2.1 Lightweight Ontologies and the Semantic Spectrum

We mentioned in the Introduction that ontologies were described in some logical language with an associated underlying semantics and certainly this was the view of early proponents of the application of philosophical ontology to AI (McCarthy, 1980). However, by 1999 it was apparent that the use of the term ontology in the AI community was proliferating and being applied more and more widely. A panel at that year’s AAAI conference³ reported on a wide spectrum of structuring mechanisms that had been characterised as ontologies. Figure 9.1 shows our own adaptation of that spectrum in the light of more recent developments.

Figure 9.1 depicts a number of approaches to information modelling in roughly increasing order of expressiveness which have been adopted in recent years, many of which have attracted the description “ontology.” Note that Obrst in Chapter 2 offers a similar model.

By *term list* is meant any set of terms used to denote entities in a particular domain of interest. A good contemporary example of a term list would be a tag cloud. In Web 2.0⁴ parlance, a tag is a (relevant) keyword or term associated with or assigned to a piece of content, thus describing the item and enabling keyword-based classification of information for the purpose of browsing and retrieval. A tag cloud is then a set of tags defined on a particular body of content to enable topic browsing. Each tag in the tag cloud is a link to the collection of items that have that tag. Tags are usually chosen by the author or consumer of the content and are thus not part of any shared, more formal classification scheme. Tag clouds are sometimes

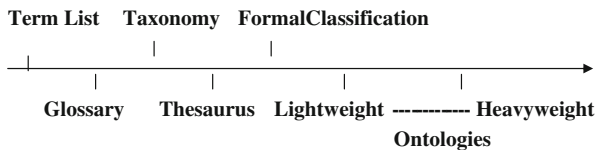


Fig. 9.1 Semantic spectrum

³<http://www.cs.vassar.edu/faculty/welty/aaai-99/>

⁴<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>

also referred to as “folksonomies” (from “folk” + “taxonomy”, though folksonomies have little in common with taxonomies, being devoid of structure). Glossaries are closely related to term lists, being term lists with associated definitions for each term.

Thesaurus is a term which, as seen earlier in the case of the term ontology, has been adapted in computer science and taken on a related but different meaning from its original sense. Originally denoting a book listing words in groups of synonyms (and sometimes also antonyms and related concepts), in computer science thesaurus typically denotes a collection of terms denoted by three relations: broader-than (BT), narrower-than (NT) and related-term (RT) sometimes augmented by further relations. Thesauri in this context are used by electronic information providers to associate terms with documents, often in digital libraries, to assist information retrieval and browsing. The semantics of thesauri relations are not always entirely clear, as we will discuss later.

A *taxonomy* is semantically more rigorous than a thesaurus in that it is comprised of a hierarchy of concepts linked by a transitive subsumption relation (often called **isA** or **subClassOf**) whereby each instance of a class can be inferred to be an instance of all parent classes. Taxonomies are strict hierarchies: each class has at most one parent.⁵

Formal classification schemes have recently been proposed (Giunchiglia et al., 2005) in an attempt to formalise previous work on classification. Classification has a long history as the discipline of grouping related concepts or entities. Its task is to aggregate items for some specific purpose, in contrast to ontology, where the goal is generate a model of some domain of interest. Giunchiglia (2005) describes the representation of classifications as lightweight ontologies, as discussed later.

As mentioned earlier, ontologies have been defined, in our sense of the word, as “specifications of conceptualisations.” Borst (1997) extended this definition somewhat:

An ontology is a formal, explicit, specification of a shared conceptualisation.

Similarly, in perhaps the best definition we have encountered, Guarino and Giaretta (1995) offered the following:

a logical theory which gives an explicit, partial account of a conceptualisation.

This definition emphasises the requirement for a logical theory and that any such account will always be partial rather than being able to completely specify the intended meaning of any conceptual element.

Our own view is that to qualify as such, an ontology should offer a formal semantic account of statements in the ontological language: that is, they should be *logic-based*.⁶ Indeed, without formal semantics, it is hard to see how an ontology

⁵Taxonomy is also sometimes used loosely (and incorrectly) to denote any set of categories against which electronic content has been classified.

⁶Though note that we do not necessarily exclude non-logical but formal accounts (e.g. in the case where the knowledge is probabilistic in nature).

can be sharable or re-usable, since there is no clear account of what statements in the language used to represent the ontology actually mean. In addition, the representation should be *declarative* so that ontological concepts and other features (such as the constraints imposed on their use) are explicitly defined. In addition, in order to qualify as a *conceptualisation* an ontology must be more than a set of ground facts⁷: rather it is an abstract model of concepts in the world, usually limited to a particular domain of interest.

Lightweight ontologies would then typically consist of a hierarchy of concepts and a set of relations holding between those concepts. As discussed earlier, if we specify only a hierarchy of concepts related using a subsumption relation, then we have a *taxonomy*. Conversely, *heavyweight* ontologies add cardinality constraints, standalone axioms, reified statements and more.

Depending on the planned use of the ontology, a deeper⁸ ontology may or may not be required. The deeper an ontology, the more resources it requires to construct and maintain and of course justifying the required investment in ontology creation will depend on the anticipated value of the uses(s) to which the ontology will be put.

We stated above that ontologies should be specified in a logical language and much research in recent years has focussed on the use of Description Logics for this purpose. Description Logics (DLs) are logics based on a subset of first order predicate calculus, so called because they focus on descriptions of concepts (classes) as a principal means for expressing logical propositions. A description logic system emphasises the use of classification and subsumption reasoning as its primary mode of inference. DLs were designed as an extension to frames and semantic networks, which were not equipped with formal logic-based semantics. DLs are a very natural candidate for specifying ontologies, given their focus on concepts and subsumption reasoning and the fact that, being subsets of FOPC, they have attractive computational properties. Indeed, there has been extensive research into the theoretical underpinnings of DLs and a family of logics set out, each with well-understood properties in terms of expressive power and computational tractability (Baader et al., 2003).

In recent years, the role and impact of ontologies in computer science has increased significantly with the advent of the semantic web (Fensel, 2001). In the semantic web, ontologies provide the key mechanism whereby web-based metadata is made machine-interpretable. They provide the formal, shared, explicit domain model against which web data can be annotated. The World Wide Web Consortium (W3C) has developed the OWL family of languages as open standards for specifying and using ontologies on the (semantic) web. In Section 9.6 we review the OWL variants and their relative expressivity and complexity.

In recent years, a number of researchers have constructed lightweight ontologies for use in a number of different application scenarios, often providing mechanisms

⁷The OWL web ontology language, for example, allows a set of ground facts to be defined as an ontology. Strictly speaking, this is inadmissible.

⁸‘Deeper’ in the sense more elaborate and offering a more precise model of the domain at hand.

to map between other formalisms on the semantic spectrum into a lightweight ontology as we have defined it. This work is important since, firstly, it adds to these previous schemes the benefits of rigour, formal semantics and improved machine processability and, secondly, because it provides a way of generating domain ontologies at reasonable cost based on pre-existing work typically with at least some degree of community consensus.

In the remainder of this section, we will give examples of methods of converting schemes from a number of points lower on our semantic spectrum, before proceeding in the rest of the chapter for a more detailed discussion of lightweight ontologies on the semantic web.

9.2.2 *Folksonomies and Lightweight Ontologies*

Motivated by the wide use of folksonomies on the one hand and the benefits of the more formal, consistent ontological approach on the other, Van Damme et al. (2007, 2008) describe a method for “turning folksonomies into ontologies.”

As described above, aggregation of raw user-supplied metadata (tags) leads to a tag cloud or folksonomy, as exemplified in systems such as Flickr⁹ or del.icio.us.¹⁰ Problems with the use of unstructured, uncontrolled tag clouds include:

- (i) different tags referring to the same concept (“Mr Bush” “George Bush” “the president”);
- (ii) the same tag referring to different concepts (“bank” referring to a financial institution or referring to an area of sloping land, for example along a riverside);
- (iii) different users tagging the same content at different conceptual levels of abstraction (“Eiffel Tower” or “Paris”).

Building on previous work including Specia and Motta (2007), Van Damme and her co-authors propose five sets of resources available for deriving ontologies from folksonomies:

- (i) the statistical analysis of both folksonomies and the usage of folksonomy-based systems (including the underlying social relationships between users) to identify structural patterns in folksonomies;
- (ii) available lexical resources such as dictionaries, Wordnet and Wikipedia;
- (iii) existing ontologies
- (iv) tools for ontology mapping and matching
- (v) methodologies for assisting a community (of ontology-builders in a given domain) in finding and maintaining consensus.

The contribution of each of the above type of resource to the process is discussed and analysed.

⁹<http://www.flickr.com/>

¹⁰<http://del.icio.us/>

9.2.3 Thesauri and Lightweight Ontologies

Thesauri are controlled vocabularies developed in specific domains for the purpose of annotating electronic content and exploiting such annotations for enhanced information retrieval and browsing. Most thesauri are represented in special purpose data formats and often use relations between categories whose semantics are not well-defined. Thus converting thesauri to lightweight ontologies, as is in the case folksonomies, brings a number of benefits:

- adherence to a open standard and the consequent enhanced interoperability
- a formal semantic basis
- an explicit specification
- a machine-processable representation

Assem et al. (2006) present a method for the conversion of thesauri to a lightweight ontology SKOS (Miles and Brickley, 2005). Essentially, SKOS is used as a metamodel for representing thesauri in RDF. Three specific cases are analysed, revealing that two of the cases have non-standard features not (currently) anticipated by the method presented. Nevertheless, it is concluded that the metamodel does seem applicable for representing resources adhering to the ISO2788 standard for monolingual thesauri.

9.2.4 Formal Classification and Lightweight Ontologies

As mentioned above Giunchigla et al. (2005) introduces the notion of *formal classification*. Noting that human-crafted classifications (e.g. DMOZ,¹¹ Dewey Decimal Classification¹²) lack a key ontological property, namely representation in a formal language over which automated reasoning can be performed, formal classification is developed as a graph structure where labels are written in a formal concept language. It is shown that formal classifications are equivalent to a form of lightweight ontology. The notion of Normalized Formal Classification (NFC) is developed. An NFC is an FC wherein labels of child nodes are always more specific than the labels of their parent nodes. A fully automated method is then presented for document classification into NFCs using propositional reasoning.

9.3 Ontologies and the Semantic Web

The principal application area for lightweight ontologies in Computer Science today is in the application of semantic technology to a number of application areas. This smenatic technology has been developed as part of the W3C's semantic web

¹¹<http://www.dmoz.org/>

¹²<http://www.oclc.org/dewey/>

initiative, which is described in this section. The term semantic technology is a broader term, used to denote the application of semantic web technology both on the web and in other areas.

Despite its explosive growth over the last decade, the Web remains essentially a tool to allow humans to access information. The next generation of the web, dubbed “the Semantic Web,” will extend the web’s capability through the increased availability *machine-processable* information.

Currently, web-based information is based primarily on documents written in HTML, a language useful for describing the visual presentation of webpages through a browser. HTML and today’s web, however, offer only very limited ways of describing the *content* itself. So, for example, you can specify that a given string should be displayed in a large bold font but you cannot specify that the string represents a product code or product price.

Semantic Web technology aims to address this shortcoming, using the descriptive languages RDF and OWL, and the data-centric, customizable markup language XML. These technologies, which are standards of the W3C¹³ (WorldWideWeb Consortium), allow rich descriptions of the content of Web documents. These machine-processable descriptions in turn allow more intelligent software systems to be written, automating the analysis and exploitation of web-based information.

In this paper, we begin by describing the key technology building blocks of the semantic web, namely the languages XML and RDF and the notion of ontologies. We then proceed to discuss the application of this technology in the key application area of eBusiness through the use of semantic web services.

Underpinning the Semantic Web is a stack of languages, often drawn in a Fig. 9.2 first presented by Berners-Lee in a presentation to the XML 2000 conference¹⁴:

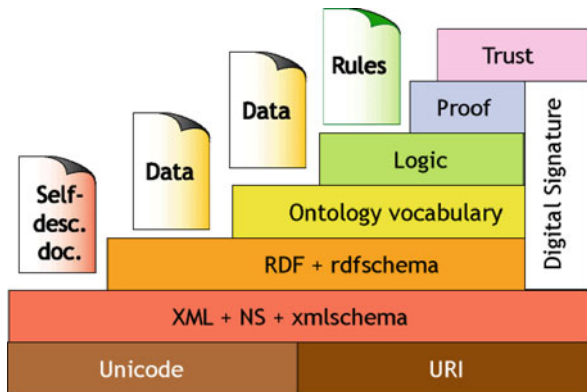


Fig. 9.2 Semantic web technology layers

¹³<http://www.w3c.org/>

¹⁴<http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide1-0.html>

We will briefly discuss the XML, RDF and Ontology vocabulary layers in this language-stack before describing ontologies and their use in the semantic web.

XML is already a widely-used language designed for annotating documents of arbitrary structure with information concerning the *content* of those documents, as opposed to HTML, which was designed principally for information *presentation* as described in the introduction. A well-formed XML document creates a balanced tree of nested sets of open and close tags. There is no fixed tag vocabulary or set of allowable combinations, so these can be defined for each application. One set of XML documents might use tags such as <price> and <quantity> and <delivery-date>, while another might use <author>, <title> and <abstract>.

Perhaps the most important use of XML is as a uniform data-exchange format. An XML document can essentially be transferred as a data object between two applications.

As mentioned above, XML is a useful language for attempting to define data exchange formats, particularly when building new interoperable systems from scratch. However, new systems will frequently need to interact with pre-existing systems. Different pre-existing systems will very often use the same term for different concepts. These types of conflicts typically require more extensive *semantics-based* solutions.

Figure 9.3 below shows two examples of semantic conflicts that can found across data sets. These conflicts are very frequent, occurring as a natural consequence of data modeling – whether due to isolated development, changing needs, organizational or structural differences, or simply the different approach of 2 human data modellers.

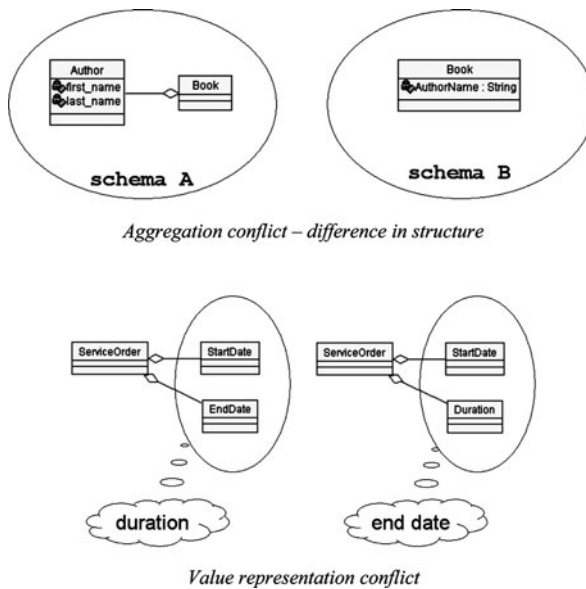


Fig. 9.3 Types of semantic conflicts (adapted from Pollock and Hodgson, 2004)

It is apparent that XML only partially addresses the data interoperability issue. We need interoperation between the processes and information sets within and across organizations without costly point-to-point data and terminology mappings. The goal of semantic web technologies is to employ logical languages that expose the structures and meanings of data more explicitly, thereby allowing software to interact whether terms and definitions are equivalent, different, or even contradictory. This is the goal of the RDF and OWL languages.

XML can be seen as *prescriptive*, in that it pre-defines the format of a data object, RDF is *descriptive*, allowing the description of semantic relationships between web resources, through the use of <subject, verb, object> triples. Sets of these triples can also be viewed as creating a graph structure. This graph structure in effect creates a web of *meaning*, where links between web resources have a semantic element (in contrast to today’s web, where pages may be linked but it is for a human user to interpret why they are linked and the relationship between the information they contain).

For example, the graph structure in Fig. 9.4 expresses the following three triples:

1. “<http://www.famousauthor.org/id21>”, hasName, “J Tolkien”>
2. “<http://www.famousauthor.org/id21>”, authorOf, “<http://www.books.org/ISBN00615861>”
3. “<http://www.books.org/ISBN00615861>”, hasPrice, “£39”>

Notice how the RDF triples allow us to combine references to web resources with literal values (e.g. character strings or numbers). Triple 1 above, for example, roughly speaking states that the famous author described at web location “<http://www.famousauthor.org/id21>” has the name “J Tolkien”.

RDF¹⁵ allows us to build a concept map (or ontology) of our domain, defining the key classes or concepts (authors, books, etc) and the relationships between those concepts (e.g. authors write books). RDF then allows us to represent and reason about specific instances of those concepts (e.g. “J. Tolkien wrote the book described at <http://www.books.org/ISBN00615861>”).

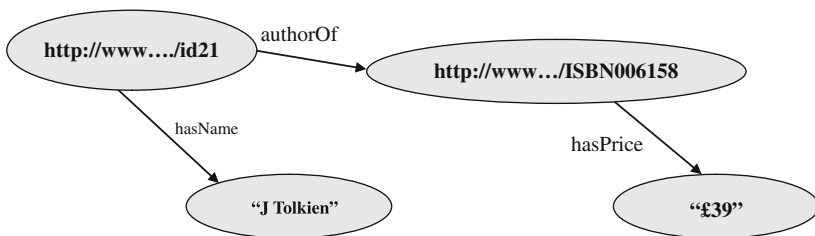


Fig. 9.4 RDF triples as a graph

¹⁵Strictly, RDF and its sister language RDF Schema

Crucially, RDF Schema includes a number of primitives which have precisely defined semantics. This means that, unlike in XML, the meaning of these primitives are formally defined and hence known to any application.

The OWL Web Ontology Language¹⁶ is a language for the “Ontology vocabulary layer of the semantic web layer cake (Fig. 9.2) and is, like XML and RDF, a standard of the W3C. Ontologies written in OWL facilitate greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

In a nutshell, RDF and OWL provide a framework for establishing relationships between data and for specifying constraints on different data elements and their relationship to one another. Furthermore, they allow for a description of pre-existing web-based data rather than prescribing a rigid format for web pages as in XML. This means that data on the web becomes machine-processable as well as human readable. In turn, this allows machines to do more of the hard work of interpreting data which today is left to the user.

In the next sections, we discuss the applications of semantic technology (and hence lightweight ontologies) to information integration, knowledge management, and to service-oriented environments.

9.4 Ontologies and Information Integration

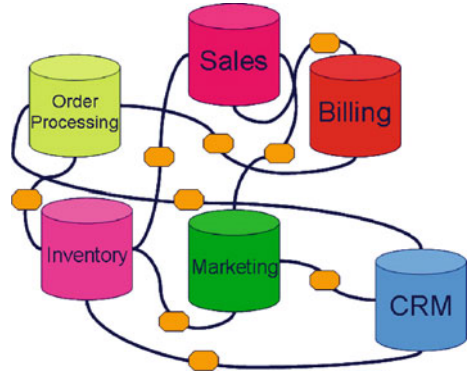
Modern organisations are characterised by the availability of massive volumes of information, made possible by electronic technology. The ability to find and share information rapidly and effectively is a clear commercial advantage in all sectors. So, too, is the ability to retain information. Maintaining the corporate memory bank as employees leave the company is an oft discussed issue.

To these commercial issues have been added regulatory ones. Organisations which do not disclose all relevant information to regulatory authorities may be seriously penalised. Yet the organisation can only disclose information it knows it has. Information lost on corporate computers can not be disclosed at the appropriate time; but will certainly be revealed if the organisation is subject to a detailed forensic analysis of hard drives prior to a legal hearing.

A typical large organisation in an information-intensive sector (such as finance) will have a number of data silos (e.g. an HR system, a CRM system, one or more billing systems, etc). Each such system has its own data model, no two of which are typically the same, making exchange and integration of information difficult: indeed, analysts report that the majority of businesses resort to expensive new software and/or manual processes when confronted with a need to integrate information from multiple sources.

¹⁶<http://www.w3.org/TR/owl-guide/>

Fig. 9.5 Information integration



The value of ontologies in information integration stems from the ability to create an over-arching ontology which can subsume multiple database schemas. The current state-of-the-art in information integration is represented in Fig. 9.5. To achieve integration at the semantic level, mappings are created between each level. These might be databases internal to one organisation, e.g. order processing and stock control databases; or the mappings might be across organisations, e.g. between databases held by separate companies working together in a joint venture or supply chain. In any case, the problem is that the number of mappings increases quadratically with the number of databases.

Ontologies can help to address this issue by providing a uniform access layer to heterogeneous data sources: the linkage of multiple structured, semi-structured and unstructured information sources using a consistent vocabulary makes it easier to build applications pulling data together from across the enterprise and also facilitates the introduction of new systems and databases (Fig. 9.6).

Advantages of the semantic integration approach as opposed to others include:

- no need to re-engineer legacy data sources – existing data sources are instead “wrapped” by a semantic description;
- based on lightweight, open standards from W3C;
- inherently extensible – RDF and OWL have been designed to make it relatively straightforward to integrate concepts and relations from more than one ontology;

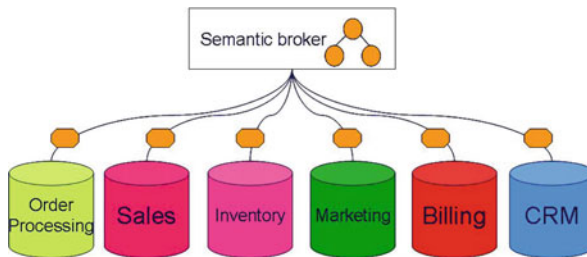


Fig. 9.6 Semantic information integration

- reasoning with the information – because OWL (for example) is a logical language, formal reasoning is supported, allowing the inference of new facts from the explicitly stored information and allowing the definition of business rules.

We have reduced the number of mappings needed, but they still do have to be created. One way to create mappings is to use a mapping language. This is fine for specialist knowledge engineers but others need a more natural and intuitive approach which is easy to learn and use. A number of graphical mapping tools have been created for such users. One such has been developed as part of their OntoStudio ontology engineering environment.¹⁷

Simple drag-and-drop functionality is used to create and amend mappings. At the same time, the system undertakes consistency checks to ensure that the user’s actions make sense. Figure 9.7 shows a view of the mapping tool. The left and right-hand side shows portions of two different ontologies, and the mappings are represented by lines between them. Mappings can even be conditional. Consider,

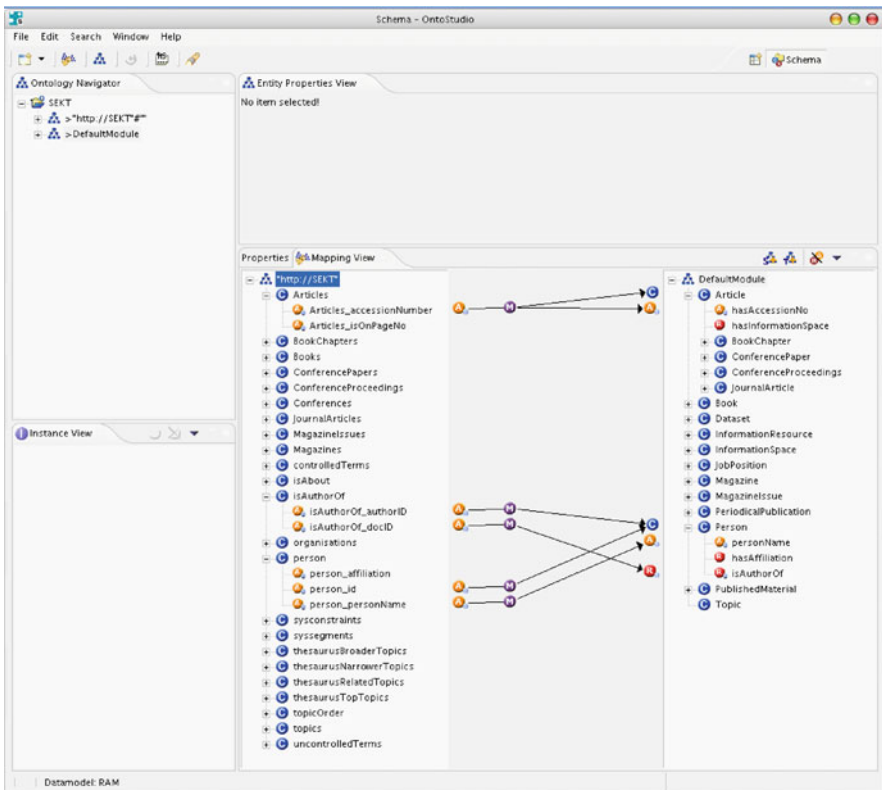


Fig. 9.7 Ontology mapping tool

¹⁷<http://www.ontoprise.de/>

for example, a mapping between two national transport ontologies. The definition of a “truck” differs in different countries, depending in some countries on the weight of the vehicle. This can be taken into account when creating the mapping.

Even greater gains can be achieved by automating, at least partially, the process of creating the mappings. This is an area of current research. A starting approach is to look for similarities in the text strings used to denote data fields by different schemas, e.g. *phone* for *telephone*. We can even take account of different representations of similar sounds, e.g. the use of *4* to represent *for*. This is frequently called syntactic matching. We can introduce some appreciation of semantics by using a thesaurus, such as Wordnet,¹⁸ to identify synonyms. Semantic matching can go further by taking account of the structure inherent in two schemas. For example, a product classification system can in general be represented as a graph.¹⁹ Structural similarities then enable the software to draw reasonable conclusions about the relationship between nodes (i.e. categories of products) in two classification systems. The software may propose equivalences between categories, or that a category in one system is a subset of a category in the other classification. Readers interested in the technical detail of one approach, based on the use of a form of logic known as propositional calculus, are referred to Bouquet et al. (2003).

Once these techniques have been used to create an initial mapping, it can then be loaded into a graphical editing tool and refined manually.

The end result is that it is possible to integrate heterogeneous databases, and provide the knowledge worker in an organisation with a unified view across these databases.

9.5 Ontologies and Knowledge Management

We have seen in the previous section how semantic technology can be applied to the integration of structured information. At least as pressing an issue, and more technically challenging, is management of *unstructured* information, part of the knowledge management problem.²⁰

¹⁸Wordnet is a lexical reference system in which English nouns, verbs, adjectives and adverbs are organised into synonym sets, with relations linking the sets. Wordnet provides both a web-based user interface for the casual user and also a programming interface to enable incorporation into other systems, e.g. software for mapping between different terminologies.

¹⁹In general a graph, but frequently a tree where the product classification is organised as a strict hierarchy.

²⁰It is not our intention to discuss the definition of knowledge management here but a broad definition could be “the management by an organisation of its intellectual assets to deliver more efficient and effective ways of working.” Better access to and management of unstructured information is a key part of this endeavour.

9.5.1 Limitations of Current Technology

The traditional approach is to provide tools (in particular search engines) based on text-string matching. This may be simply through a user initiating a search, or through text searches embedded in an application. In any case, there are several problems with this approach, which can be divided into four main areas:

(i) *Query Construction*

In general, when specifying a search, users enter a small number of terms in the query. Yet the query describes the information need, and is commonly based on the words that people expect to occur in the types of document they seek. This gives rise to a fundamental problem, in that not all documents will use the same words to refer to the same concept. Therefore, not all the documents that discuss the concept will be retrieved by a simple keyword-based search. Furthermore, query terms may of course have multiple meanings (query term *polysemy*). As conventional search engines cannot interpret the sense of the user's search, the ambiguity of the query leads to the retrieval of irrelevant information.

Although the problems of query ambiguity can be overcome to some degree, for example by careful choice of additional query terms, there is evidence to suggest that many people may not be prepared to do this. For example, an analysis of the transaction logs of the Excite WWW search engine (Jansen et al., 2000) showed that web search engine queries contain on average 2.2 terms. Comparable user behaviour can also be observed on corporate Intranets. An analysis of the queries submitted to BT's Intranet search engine over a 4-month period between January 2004 and May 2004 showed that 99% of the submitted queries only contained a single phrase and that, on average, each phrase contained 1.82 keywords.

(ii) *Lack of Semantics*

Converse to the problem of polysemy, is the fact that conventional search engines that match query terms against a keyword-based index will fail to match relevant information when the keywords used in the query are different from those used in the index, despite having the same meaning (index term *synonymy*). Although this problem can be overcome to some extent through thesaurus-based expansion of the query, the resultant increased level of document recall may result in the search engine returning too many results for the user to be able to process realistically.

In addition to an inability to handle synonymy and polysemy, conventional search engines are unaware of any other semantic links between concepts. Consider for example, the following query:

“telecom company” Europe “John Smith” director

The user might require, for example, documents concerning a telecom company in Europe, a person called John Smith, and a board appointment. Note, however, that a document containing the following sentence would not be returned using conventional search techniques:

At its meeting on the 10th of May, the board of London-based O2 appointed John Smith as CTO

In order to be able to return this document, the search engine would need to be aware of the following semantic relations:

O2 is a mobile operator, which is a kind of telecom company;
 London is located in the UK, which is a part of Europe;
 A CTO is a kind of director.

(iii) *Lack of Context*

Many search engines fail to take into consideration aspects of the user's context to help disambiguate their queries. User context would include information such as a person's role, department, experience, interests, project work, etc. A simple search on BT's Intranet demonstrates this. A person working in a particular BT line of business searching for information on their corporate clothing entitlement is presented with numerous irrelevant results if they simply enter the query "corporate clothing". More relevant results are only returned should the user modify their query to include further search terms to indicate the part of the business in which they work. As discussed above, users are in general unwilling to do this.

(iv) *Presentation of Results*

The results returned from a conventional search engine are usually presented to the user as a simple ranked list. The sheer number of results returned from a basic keyword search means that results navigation can be difficult and time consuming. Generally, the user has to make a decision on whether to view the target page based upon information contained in a brief result fragment. A survey of user behaviour on BT's intranet suggests that most users will not view beyond the 10th result in a list of retrieved documents. Only 17% of searches resulted in a user viewing more than the first page of results.²¹ Essentially, we would like to move from a document-centric view to a more knowledge-centric one (for example, by presenting the user with a digest of information gleaned from the most relevant results found as has been done in the Squirrel semantic search engine described later in this chapter).

In recent years, considerable effort has been put into the use of ontologies to deal with the problem of managing and accessing unstructured information and we summarise some of the key aspects in the remainder of this section.

²¹Out of a total of 143,726 queries submitted to the search engine, there were 251,192 occasions where a user clicked to view more than the first page of results. Ten results per page are returned by default.

9.5.2 Applying Ontologies in Knowledge Management

The essential challenge is to create some (meaningful) structure out of unstructured text. One way to do this is to create semantic *metadata*: data describing the unstructured text.

Such metadata can exist at two levels. They can provide information about a document or a page, e.g. its author, creation or last amendment date, or subject area (topic); or they can provide information about entities in the document, e.g. the fact that a particular string in the document represents a company, or a person or a product code. The metadata themselves should describe the document or entities within the document in terms of an ontology. At the document level we might have a property in the ontology *has Author* to describe authorship. Within the document we would use classes such as Person, Company or Country to identify specific entities.

Such metadata could be created by the authors of the document. In general this will not happen. The authors of Word documents or emails will not pause to create metadata. We need to generate metadata automatically, or at least semi-automatically.

There are two broad categories of technology which we can use for this: statistical or machine learning techniques²²; and information extraction techniques based on natural language processing. The former generally operate at the level of documents, by treating each document as a “bag of words”. They are, therefore, generally used to create metadata to describe documents. The latter are used to analyse the syntax of a text to create metadata for entities within the text, e.g. to identify entities as Persons, Companies, Countries etc. Nevertheless, this division should not be seen too starkly. For example, one of the goals of the SEKT project (<http://www.sekt-project.com>) was to identify the synergies which arise when these two different technologies are used closely together. An overview of semantic knowledge management, including these two approaches to creating metadata, is given in Davies et al. (2005). For more detail, see Davies, Studer and Warren (2005), which contains a chapter on each of these approaches, besides information on a number of other topics discussed in this paper.

The metadata can create a link between the textual information in the documents and concepts in the ontology. Metadata can also be used to create a link between the information in the document and instances of the concepts. This process is known as semantic annotation.

To give an example of the linkage between documents and the ontology and knowledgebase, we can imagine that the ontology will contain the concept Company. Then the text string “BT” in the document will be identified as being an instance of the concept Company. This is made possible by natural language

²²Statistical techniques employ algorithms with well-defined mathematical properties, usually derived based on certain assumptions about the datasets being used. Machine learning techniques are generally heuristic techniques with no formally derived mathematical properties, e.g. iterative techniques for which no proof of convergence exists. The two approaches may suit different circumstances, or can be used together in a complementary fashion.

processing software which can make intelligent deductions about what kind of entities particular text strings represent. For example, the software can make inferences from the way the term “BT” is used, and the words associated with it in the text. If the company British Telecommunications Plc. (to give it its formal name) exists as an instance of Company in the knowledgebase, then the software will tag “BT” as referring to British Telecommunications Plc. This is possible through an understanding of how acronyms are formed. In other situations it may even be able to tag “it” as referring to the same company, on the basis of a relatively shallow analysis of the text.²³

Where the system identifies a text string as an instance of a concept in the ontology but which is not represented in the knowledgebase, then that instance can be added to the knowledgebase. For example, the text string “ABC Holdings” may be identified as a company, but one not represented in the knowledgebase. The system can then add “ABC Holdings” to the knowledgebase.

Figure 9.8 illustrates part of an ontology and corresponding knowledgebase, and shows how entities in the text can be associated with entities in the knowledgebase.

Research is also in progress to use natural language processing techniques to learn concepts from text, and thereby extend the ontology. However, this is a significantly harder problem. For an example of the state of the art, see Cimiano and Völker (2005).

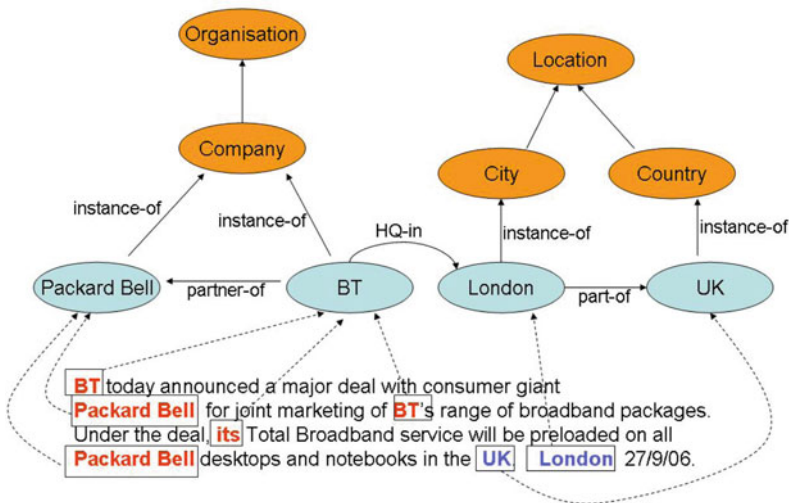


Fig. 9.8 Semantic annotation

²³The use of pronouns and other short words in place of longer words or phrases is called anaphora. Hence, the matching of such short words with their longer equivalent is called *anaphora resolution*.

9.5.3 Semantic Knowledge Management Tools

The ontological approach offers significant advantages in the capability of tools for the management of unstructured text. In this section, we exemplify with one such tool. Bontcheva et al. (2006) is a more comprehensive discussion of semantic information access discussing a wider range of tools. Similarly Mangrove (2007) contains a survey and classification of ontology-based search technology.

9.5.3.1 Squirrel Semantic Search Engine

Squirrel (Duke et al., 2007) provides combined keyword based and semantic searching. The intention is to provide a balance between the speed and ease of use of simple free text search and the power of semantic search. In addition, the ontological approach provides the user with a rich browsing experience. Squirrel builds on and integrates a number of semantic technology components:

- (i) PROTON,²⁴ a lightweight ontology and world knowledge base is used against which to semantically annotate documents.
- (ii) Lucene²⁵ is used for full-text indexing;
- (iii) The KAON2 (Motik and Studer, 2005) ontology management and inference engine provides an API for the management of OWL-DL and an inference engine for answering conjunctive queries expressed using the SPARQL²⁶ syntax. KAON2 also supports the Description Logic-safe subset of the Semantic Web Rule Language²⁷ (SWRL). This allows knowledge to be presented against concepts that goes beyond that provided by the structure of the ontology. For example, one of the attributes displayed in the document presentation is “Organisation”. This is not an attribute of a document in the PROTON ontology; however, affiliation is an attribute of the Author concept and has the range “Organisation”. As a result, a rule was introduced into the ontology to infer that the organisation responsible for a document is the affiliation of its lead author;
- (iv) OntoSum (Bontcheva, 2005) a Natural Language Generation (NLG) tool, takes structured data in a knowledge base (ontology and associated instances) as input and produces natural language text, tailored to the presentational context and the target reader. In the context of the semantic web and knowledge management, NLG is required to provide automated documentation of ontologies and knowledge bases and to present structured information in a user-friendly way;
- (v) KIM (Popov et al., 2003) is used for massive semantic annotation.

²⁴<http://proton.semanticweb.org/>

²⁵<http://lucene.apache.org/>

²⁶<http://www.w3.org/TR/rdf-sparql-query/>

²⁷<http://www.w3.org/Submission/SWRL/>

Initial Search

Users are permitted to enter terms into a text box to commence their search. This initially simple approach was chosen since users are likely to be comfortable with it due to experience with traditional search engines. Squirrel then calls the Lucene index and KAON2 to identify relevant textual resources or ontological entities, respectively. In addition to instance data, the labels of ontological classes are also indexed. This allows users to discover classes and then discover the corresponding instances and the documents associated with them without knowing the names of any instances e.g. a search for “Airline Industry” would match the “Airline” class in PROTON. Selecting this would then allow the user to browse to instances of the class where they can then navigate to the documents where those instances are mentioned.

Meta-Result

The meta-result page is intended to allow the user to quickly focus their search as required and to disambiguate their query if appropriate. The page presents the different types of result that have been found and how many of each type.

The meta-result for the “home health care” query is shown in Fig. 9.9 under the sub-heading “Matches for your query”.

Document View

The user can select a document from the result set, which takes them to a view of the document itself. This shows the meta-data and text associated with the document and also a link to the source page if appropriate – as is the case with web-pages. Semantically annotated text (e.g. recognised entities) are highlighted. A screenshot of the document view is shown in Fig. 9.10.

“Mousing-over” recognised entities provides the user with further information about the entity extracted from the ontology. Clicking on the entity itself takes the user to the entity view.



Fig. 9.9 Meta-result

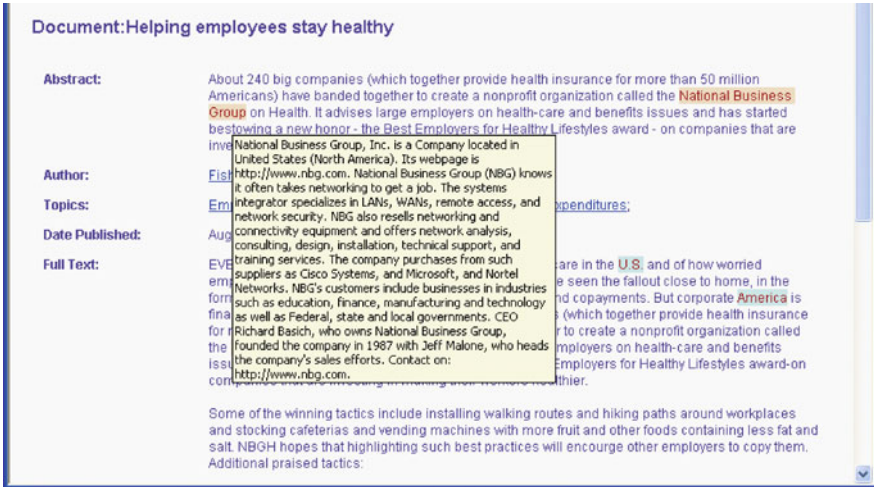


Fig. 9.10 Document view

Entity View

The entity view for “Sun Microsystems” is shown in Fig. 9.11. It includes a summary generated by OntoSum. The summary displays information related not only to the entity itself but also information about related entities such as people who hold job roles with the company. This avoids users having to browse around the various entities in the ontology that hold relevant information about the entity in question.

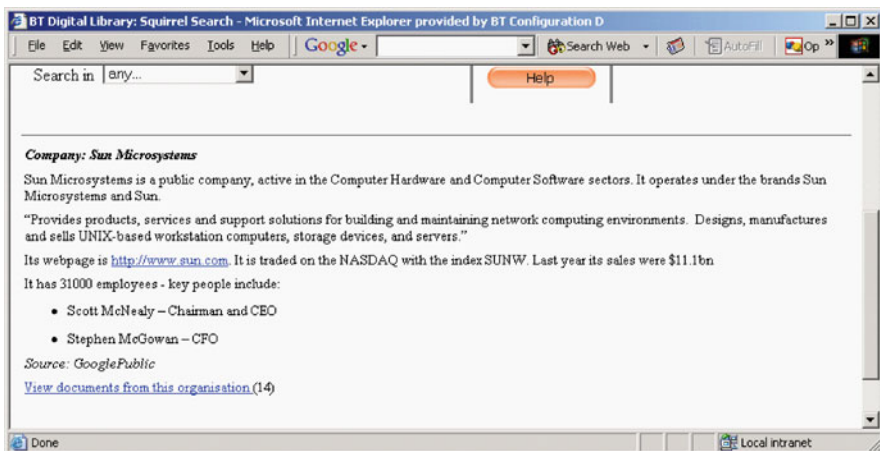


Fig. 9.11 Company entity view

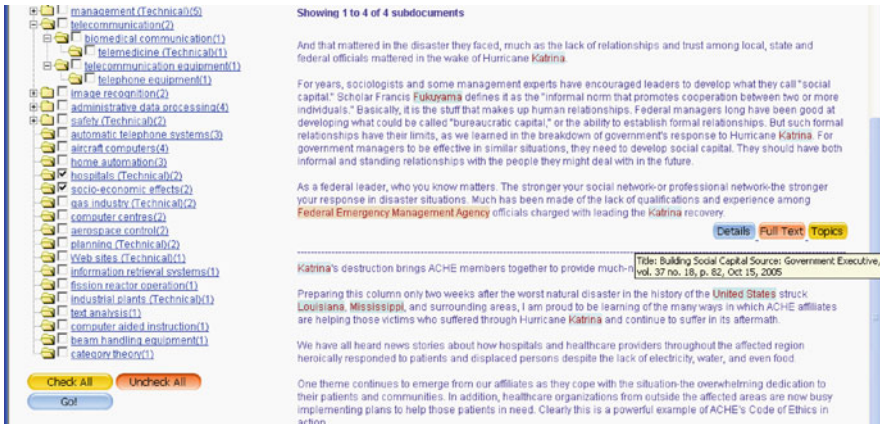


Fig. 9.12 Consolidated results

Consolidated Results

Users can choose to view results as a *consolidated summary* (digest) of the most relevant parts of documents rather than a discrete list of results. The view allows users to read or scan the material without having to navigate to multiple results. Figure 9.12 shows a screenshot of a summary for a query for “Hurricane Katrina”. For each subdocument in the summary the user is able to view the title and source of the parent document, the topics into which the subdocument text has been classified or navigate to the full text of the document. Evaluation

Squirrel has been subjected to a three-stage user-centred evaluation process with users of a large Digital Library. Results are promising regarding the perceived information quality (PIQ) of search results obtained by the subjects. From 20 subjects, using a 7 point scale the average (PIQ) using the existing library system was 3.99 vs. an average of 4.47 using Squirrel – an 12% increase. The evaluation also showed that users rate the application positively and believe that it has attractive properties. Further details can be found in Thurlow and Warren (2008).

9.6 Ontologies and Service-Oriented Environments

Industry is seeking urgently to reduce IT costs, more than 30% of which are attributable to integration.²⁸ Furthermore, in the telecommunications sector for example, costs of OSS²⁹ integration can rise to 70% of the total OSS budget.³⁰

²⁸Gartner Group, 2004.

²⁹Operational Support Systems: systems that support the daily operation of an organisation’s business including, for example, billing, ordering, delivery, customer support.

³⁰See, for example, http://www.findarticles.com/p/articles/mi_m0TLC/is_5_36/ai_86708476

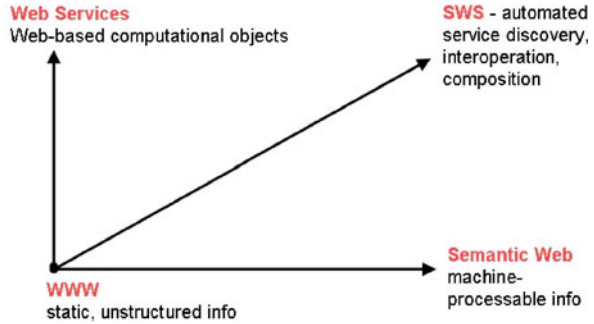
In addition, there is a need to reconfigure system components efficiently in order to satisfy regulatory requirements for interoperability and to respond quickly to increasingly sophisticated customer requirements for bundled services.

Thus one of the most pressing current issues design of software architectures is to satisfy increasing software complexity as well as new IT needs, such as the need to respond quickly to new requirements of businesses, the need to continually reduce the cost of IT or the ability to integrate legacy and new emerging business information systems. In the current IT enterprise settings, introducing a new product or service and integrating multiple services and systems present unpredicted costs, delays and difficulty. Existing IT systems consist of a patchwork of legacy products, monolithic off-shelf applications and proprietary integration. It is even today's reality that in many cases users on the "spinning chairs" manually re-enter data from one system to another within the same organization. The past and existing efforts in Enterprise Application Integration (EAI) don't represent successful and flexible solutions. Several studies showed that the EAI projects are lengthy and the majority of these efforts are late and over budget. It is mainly costs, proprietary solutions and tightly-coupled interfaces that make EAI expensive and inflexible.

Service Oriented Architecture (SOA) solutions are the next evolutionary step in software architectures. SOA is an IT architecture in which functions are defined as independent services with well-defined, invocable interfaces. SOA will enable cost-effective integration as well as bring flexibility to business processes. In line with SOA principles, several standards have been developed and are currently emerging in IT environments. In particular, Web Services technology provides means to publish services in a UDDI registry, describing their interfaces using the Web Service Description Language (WSDL) and exchanging requests and messages over a network using SOAP protocol. The Business Process Execution Language (BPEL) allows composition of services into complex processes as well as their execution. Although Web services technologies around UDDI, SOAP and WSDL have added a new value to the current IT environments in regards to the integration of distributed software components using web standards, they cover mainly characteristics of syntactic interoperability. With respect to a large number of services that will exist in IT environments in the inter and intra enterprise integration settings based on SOA, the problems of service discovery or selection of the best services conforming user's needs, as well as resolving heterogeneity in services capabilities and interfaces will again be a lengthy and costly process. For this reason, *machine processable semantics* should be used for describing services in order to allow total or partial automation of tasks such as discovery, selection, composition, mediation, invocation and monitoring of services. As discussed, the way to provide such semantics is through the use of ontologies.

Web services technology effectively added computational objects to the static information of yesterday's Web and as such offers a distributed services capability over a network. Web services provide an easy way to make existing (or indeed new) software components available to applications via the Internet. As explained above, however, web services are essentially described using semi-structured natural language mechanisms, which means that considerable human intervention is needed to

Fig. 9.13 Web services and the semantic web



find and combine web services into an end application. As the number of services available grows, the issue of scalability becomes critical: without richer (machine-processable, semantic) descriptions of services it becomes impossible to discover and compose services in an efficient manner.

The Semantic Web enables the accessing of web resources by semantic descriptions rather than just by keywords. Resources (here, web services) are defined in such a way that they can be automatically processed by machine. This will enable the realisation of Semantic Web Services, involving the automation of service discovery, acquisition, composition and monitoring.

The relationship between web service and semantic web technology is encapsulated in Fig. 9.13 below. Combining the two technology creates the next generation of web services: semantically-enabled web services with the more sophisticated capabilities described above.

9.6.1 Web Service Modeling Ontology (WSMO)

The Web Service Modeling Ontology (WSMO) is one ontology which has been developed to provide a conceptual model for the Semantic Web Services. Another approach is OWL-S: in the interests of brevity we will focus on WSMO and refer the interested reader to W3C (2004) for further information about OWL-S. In WSMO, four elements are identified as the fundamental pillars of the model: namely, *ontologies* as shared vocabularies with clearly defined semantics, *web services* as means to abstract the IT functionality provided, *goals* representing users requests referring to the problem-solving aspect of our architecture and finally *mediators* for interpretability between the various semantic descriptions.

Ontologies are used as the data model throughout WSMO, meaning that all resource descriptions as well as all data interchanged during service usage are based on ontologies. Ontologies are a widely accepted state-of-the-art knowledge representation, and have thus been identified as the central enabling technology for the Semantic Web. The extensive usage of ontologies allows semantically enabled information processing as well as support for interoperability; WSMO also supports the ontology languages defined for the Semantic Web.

Goals provide means to characterize user requests in terms of functional and non-functional requirements. For the former, a standard notion of pre and post conditions has been chosen and the later provides a predefined Ontology of generic properties. For functional aspects a standard notion of pre and post conditions has been chosen. For non-functional properties (QoS) logical expressions are used to specify the QoS values provided by the service.

Web Service descriptions specify the functionality, the means of interaction and non-functional properties (QoS) aspects provided by the Web Service. More concretely, a Web service presents:

a capability that is a functional description of a Web service. A set of constraints on the input and output of a service as well as constraints not directly affecting the input (preconditions) and output (postconditions) of the service but which need to hold before (assumptions) and after (effects) the execution of the service are part of the capability description.

interfaces that specify how the service behaves in order to achieve its functionality. A service interface consists of a choreography that describes the interface for the client-service interaction required for service consumption, and an orchestration that describes how the functionality of a Web service is achieved by aggregating other Web services.

non-functional properties that specify the QoS values provided by the service.

Non-functional properties of services or goals are modeled in a way similar to which capabilities are currently modeled in WSMO, more precisely by means of logical expressions.

Mediators provide additional procedural elements to specify further mappings that cannot directly be captured through the usage of Ontologies. Using Ontologies provides real-world semantics to our description elements as well as machine processable formal semantics through the formal language used to specify them. The concept of Mediation in WSMO addresses the handling of heterogeneities occurring between elements that shall interoperate by resolving mismatches between different used terminologies (data level), on communicative behavior between services (protocol level), and on the business process level. A WSMO Mediator connects the WSMO elements in a loosely coupled manner, and provides mediation facilities for resolving mismatches that might arise in the process of connecting different elements defined by WSMO.

9.6.2 Web Service Modeling Language (WSML)

The Web Service Modeling Language (WSML) is a language (or more accurately a family of languages) for the description of ontologies, goals, web services and mediators, based on the conceptual model of WSMO. A major goal in the development of WSML is to investigate the applicability of different formalisms, most notably Description Logics and Logic Programming, in the area of Web services. A fuller discussion of the various WSML dialects and their advantages and disadvantages can be found in Vitvar et al. (2007).

9.6.3 Web Service Modeling Execution Environment (WSMX)

The Web Service Modeling Execution Environment (WSMX) is a reference implementation of semantic service-oriented environment that is compliant with the semantic specifications of WSMO. WSMX supports semantically enabled change functions such as dynamic discovery, selection, and mediation. WSMX also implements semantically enabled control and connection functions such as service invocation and inter-operation. WSMX is an execution environment for the dynamic discovery, selection, mediation, invocation and inter-operation of the semantic web services in a reference implementation for WSMO. The development process for WSMX includes defining its conceptual model, defining the execution semantics for the environment, describing a architecture and a software design and building a working implementation.

Figure 9.14 presents the WSMX architecture and its most important components.

In terms of functionality provided, WSMX can be seen as an aggregation of the components. The central components are the Core Component, Resource Manager, Discovery, Selection, Data and Process Mediator, Communication Manager, Choreography Engine, Web Service Modeling Toolkit, and the Reasoner.

The *Core Component* is the central component of the system connecting all the other components and managing the business logic of the system. The *Resource Manager* manages the set of repositories responsible for the persistency of all the WSMO and non-WSMO related data flowing through the system. The *Discovery* component is responsible for locating services that satisfy a specific user request. A set of discovery algorithms ranging from syntactical-based to full semantically-based matching are available in WSMX for service discovery. The *Selection* component is responsible for filtering the potential services which provide the requested functionality by considering non-functional properties values and finally selecting the best service.

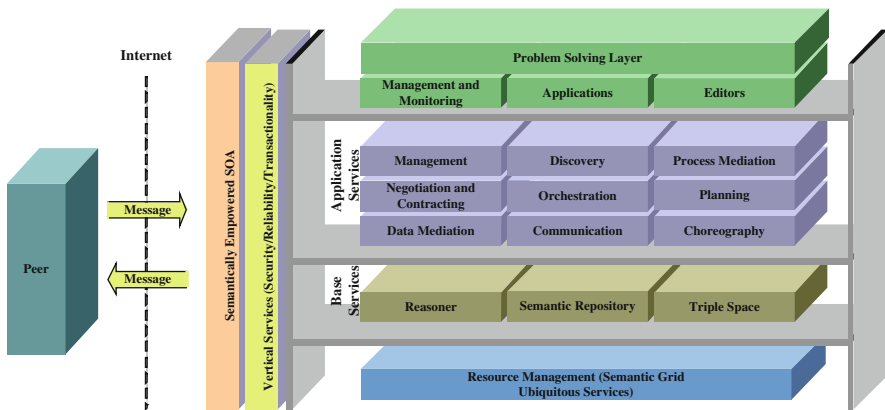


Fig. 9.14 WSMX reference architecture

WSMX provides two kinds of mediator components that deal with heterogeneity problems: a *Data Mediator* component, which mediates between different terminologies, ontologies and a *Process Mediator* component which mediates between two given patterns of interaction (i.e., WSMO choreographies) and compensates the possible mismatches that may appear. The *Communication Manager* is responsible for managing the communication between a service requester and a service provided. The two way communication is handled by the Receiver, from requester to the service, and respectively by Invoker, from the service to the requester. The *Choreography engine* provides means to store and retrieve choreography interface definitions, initiates the communication between the requester and the provider in direct correlation with the results returned by the Process Mediator, and keeps track of the communication state on both the provider and the requester sides. In addition it provides grounding information to the communication manager to enable any ordinary Web Service invocation.

The semantic inference required by each of the WSMX components is provided by the *Reasoner* component. On the client/developed side a toolkit, called the *Web Services Modeling Toolkit (WSMT)* is provided allowing modeling of WSMO elements and easy interaction with the WSMX server. WSMT contains a set of tools including the WSMO Visualizer for viewing and editing WSML documents using directed graphs, a WSMX. Management tool for managing and monitoring the WSMX environment, and a WSMX Data Mediation tool for creating mappings between WSMO ontologies are also available in WSMT.

In short, Semantic Web Services can lead to more flexible, interoperable, less costly IT systems. Space does not permit a full description of the technical details of semantic web services and the reader is referred to [Chapter 10](#) of Davies et al. (2005) for an overview of current work in Semantic Web Services and to Vitvar et al. (2007) for a discussion of the applications of semantic technology in service-centric environments.

9.7 Ontologies and Computer Science

In the above sections we have discussed some areas of applications of lightweight ontologies in IT today. Moving beyond specific application areas and taking a broader view, the central property of semantic technology is that it offers the ability to provide machine-processable descriptions. As above, these descriptions can be of documents, fragments of documents or web services. Similarly, such descriptions could equally be descriptions of grid elements, handheld computing devices, security policies or business process elements.

In all these cases, the key points are (i) these descriptions have a well-defined meaning separate from the programs which interpret them which (ii) allows *interoperability* which would otherwise require hardwired solutions handcoded by humans.

In short, semantic technology offers the only path to web-scale interoperability; and such scalability and interoperability is surely one of the most pressing research challenges for computer science today.

9.8 Conclusion

In this chapter, we have discussed the notion of lightweight ontologies. We defined ontology and related ontology to other related knowledge organisation structures, formal and less formal. We then considered what might constitute a *lightweight* ontology. Our view of lightweight as opposed to “heavyweight” ontologies centred around the expressivity of the ontology description, rather than other possible notions such as scope, depth or computational tractability. Of course, tractability is related to expressivity in that, in general, the more expressive a formal language the less it is amenable to efficient machine processing (reasoning). Regarding scope and depth, these seem to us separate and highly context-dependent issues: a medical ontology could seem very wide and/or deep in terms of its domain coverage from the point of view of a general medical practitioner and yet much shallower from an ontological consultant or rather narrow from the point of view of a science journalist interested in medicine and many other disciplines besides, for example. Scope is orthogonal to expressivity (an ontology covering a wide domain can be more or less expressive), whereas depth is in general related in the sense that an ontology requiring more expressive language to describe it will typically be deeper (i.e. model the domain of interest to a more detailed level).

We proceeded to discuss ontologies and the semantic web, the emergence of which over the past decade has seen increased interest in ontologies and associated topics. We then looked at 3 key areas where ontologies are being used in IT systems today and briefly discussed the likely centrality of ontologies and semantic technology to computer science in the future. A number of requirements need to be fulfilled for the further uptake of ontologies:

- provision of ontologies and associated metadata: clearly a barrier to the use of ontologies is the need to manually construct an ontology for each application (or at least application domain) and then to populate the ontology with instance data). The use of knowledge discovery and human language technologies is starting to address these areas, as are the ever increasing number of domain ontologies available for re-use. As a specific example of the strides being made in this area, Thomson-Reuters, the world’s largest business information provider now offers the Open Calais³¹ tool for semantically annotating large volumes of textual information, as well as providing its own information so annotated;
- production of ontology tools: there is need for mature tools supporting the ontology engineering lifecycle, preferably integrated into existing environments.

Assuming progress is made in these areas (and all are the subject of active research and development programmes), over the next decade, we can anticipate the increasing incorporation of ontology-based technology into mainstream IT systems and methods.

³¹<http://www.opencalais.com/>

Acknowledgments Céline van Damme is thanked for insightful discussions about folksonomies and informal ontologies in the enterprise. Section 9.4 is based partly on Warren and Davies (2007), while Section 9.6 is based partly on Davies et al. (2006).

References

- Alonso, O. 2006. Building semantic-based applications using Oracle. Developer's Track at WWW2006, Edinburgh. <http://www2006.org/programme/item.php?id=d16>. Accessed May 2006.
- Assem, M., V. Malaise, A. Miles, and G. Schreiber. 2006. A method to convert thesauri to SKOS. In Proceedings of the European Semantic Web Conference 2006 (ESWC 2006), Budva, Montenegro. Heidelberg: Springer.
- Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. 2003. *The description logic handbook*. Cambridge, UK: Cambridge University Press.
- Beckett, D. 2004. RDF/XML syntax specification (revised). <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>.
- Berners-Lee, T. 1999. *Weaving the web*. London: Orion Books.
- Bontcheva, K., J. Davies, A. Duke, T. Glover, N. Kings, and I. Thurlow. 2006. Semantic information access. In *Semantic web technologies: Trends and research in ontology-based systems*, eds. J. Davies, R. Studer, and P. Warren, Chichester: John Wiley.
- Brickley, D., and Guha, R.V, eds. 2000. Resource description framework (RDF) schemas, W3C. <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>.
- Bernstein, A., E. Kaufmann, A. Goehring, and C. Kiefer. 2005. Querying ontologies: A controlled english interface for end-users, Proceedings of the 4th International Semantic Web Conference, ISWC2005, Galway, Ireland, November 2005. Heidelberg: Springer.
- Borst, P., and H. Akkermans. 1997. Engineering ontologies. *International Journal of Human-Computer Studies*, 46:365–406.
- Chinchor, N., and P. Robinson. 1998. MUC-7 named entity task definition (version 3.5). In Proceedings of the 7th Message Understanding Conference (MUC-7), Fairfax, VA.
- Cunningham, H. 2000. Software architecture for language engineering. PhD Thesis, University of Sheffield.
- Cunningham, H. 1999. Information extraction: A user guide (revised version). Department of Computer Science, University of Sheffield, May 1999.
- Davies, J., D. Fensel, and F. van Harmelen. 2003. *Towards the semantic web*. Chichester: Wiley.
- Davies, J., R. Weeks, and U. Krohn. 2003. QuizRDF: Search technology for the semantic web. In *Towards the semantic web*, eds. J. Davies, D. Fensel, and F. van Harmelen. Chichester: Wiley.
- Davies, J., R. Studer, Y. Sure, and P. Warren. 2005. Next generation knowledge management. *BT Technology Journal* 23(3):175–190, July 2005.
- Davies, J., R. Studer, and P. Warren. 2006. *Semantic web technology: Trends & research*. Chichester: Wiley.
- Davies, J., et al. 2007. NESSI semantic technologies working group research roadmap 2007–2010 Version 1.2. <http://www.nessi-europe.com/Nessi/LinkClick.aspx?fileticket=Ja5zTnzK4%2FM%3d&tabid=241&mid=694>. Accessed 10th June 2007.
- DCMI Usage Board. 2005. DCMI metadata terms. <http://dublincore.org/documents/2005/06/13/dcmi-terms/>.
- Dean, M., and G. Schreiber, eds.; Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, 2004. OWL web ontology language reference. W3C recommendation. <http://www.w3.org/TR/owl-ref/>. Accessed 10 February 2004.
- Domingue, J., M. Dzbor, and E. Motta. 2004. Collaborative semantic web browsing with magpie. In *The semantic web: Research and applications*, Proceedings of ESWS, 2004, LNCS 3053, eds. J. Davies, C. Bussler, D. Fensel, and R. Studer, 388–401. Heidelberg: Springer.

- Dumais, S., E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. Robbins. 2003. Stuff I've Seen: A system for personal information retrieval and re-use. In Proceedings of SIGIR'03, Toronto. New York, NY: ACM Press.
- Dill, S., N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zienberer. 2003. A case for automated large scale semantic annotation. *Journal of Web Semantics* 1(1):115–132.
- Ding, L., T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. 2004. Swoogle: A search and metadata engine for the semantic web, Conference on Information and Knowledge Management CIKM04, Washington, DC, November 2004.
- Ehrig, M., P. Haase, M. Hefke, and N. Stojanovic. 2005. Similarity for ontologies – A comprehensive framework. In Proceedings of the 13th European Conference on Information Systems, Regensburg, May 2005.
- Fensel, D., and F. van Harmelen. 2007. Unifying reasoning and search to web scale. *IEEE Internet Computing* 11(2):94–96.
- Fensel, D., and M. Musen. 2001. The semantic web: A brain for humankind. *IEEE Intelligent Systems* 16(2):24–25.
- Giunchiglia, F., M. Marchese, and I. Zaihrayeu. 2005. Encoding classification into lightweight ontologies. In Proceedings of the 2nd European Semantic Web Conference ESWC05, Heraklion, Crete, May 2005.
- Glaser, H., H. Alani, L. Carr, S. Chapman, F. Ciravegna, A. Dingli, N. Gibbins, S. Harris, M.C. Schraefel, and N. Shadbolt. 2004. CS AKTiveSpace: Building a semantic web application. In *The semantic web: Research and applications*, Proceedings of ESWS, 2004, LNCS 3053, eds. J. Davies, C. Bussler, D. Fensel, and R. Studer, 388–401. Heidelberg: Springer.
- Grishman, R. 1997. TIPSTER architecture design document version 2.3. Technical report, DARPA. http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/.
- Gruber, T.R. 1992. A translation approach to portable ontologies. *Knowledge Acquisition* 5(2):199–220. http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html. Accessed 1993.
- Gruber, T.R. 1993. Toward principles for the design of ontologies used for knowledge sharing. In International Workshop on Formal Ontology, Padova, Italy, eds. N. Guarino and R. Poli. http://ksl-web.stanford.edu/KSL_Abstracts/KSL-93-04.html.
- Guarino, N., and P. Giaretta. 1995. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards very large knowledge bases: Knowledge building and knowledge sharing*, ed. N. Mars, 25–32. Amsterdam: IOS Press.
- Guarino, N. 1998. Formal ontology in information systems. In Proceedings of FOIS'98, Trento, Italy, 6–8 June 1998, ed. N. Guarino, 3–15. Amsterdam: IOS Press.
- Guha, R., and R. McCool. 2003. Tap: A semantic web platform. *Computer Networks* 42:557–577.
- Guha, R., R. McCool, and E. Miller. 2003. Semantic search. In WWW2003, Budapest, Hungary, 20–24 May 2003.
- Iosif, V., P. Mika, R. Larsson, and H. Akkermans. 2003. Field experimenting with semantic web tools in a virtual organisation. In *Towards the semantic web*, eds. J. Davies, D. Fensel, and F. van Harmelen. Chichester: Wiley.
- Jansen, B.J., A. Spink, and T. Saracevic. 2000. Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management* 36(2): 207–227.
- Kiryakov, A. 2006. Ontologies for knowledge management. In *Semantic web technologies: Trends and research in ontology-based systems*, Chapter 7, eds. J. Davies, R. Studer, and P. Warren. Chichester: Wiley.
- Kiryakov A., and K.Iv. Simov. 1999. Ontologically supported semantic matching. In Proceedings of the “NODALIDA'99: Nordic Conference on Computational Linguistics”, Trondheim, 9–10 Dec 1999.
- Kiryakov, A., B. Popov, D. Ognyanov, D. Manov, A. Kirilov, and M. Goranov. 2004. Semantic annotation, indexing, and retrieval. *Elsevier's Journal of Web Semantics* 1, ISWC2003 special issue (2). <http://www.websemanticsjournal.org/>.

- Kiryakov, A., D. Ognyanov, and D. Manov. 2005. OWLIM – A pragmatic semantic repository for OWL. In Proceedings of International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE, 20 Nov 2005, New York City.
- Klyne, G., and J.J. Carroll. 2004. Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation. <http://www.w3.org/TR/rdf-concepts/>. Accessed 10 Feb 2004.
- Landauer T., and S. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review* 104(2):211–240.
- McCarthy, J. 1980. Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence* 13:27–39.
- Mahesh, K., J. Kud, and P. Dixon. 1999. Oracle at TREC8: A lexical approach. In Proceedings of the 8th Text Retrieval Conference (TREC-8) Gaithersburg, Maryland.
- Mangold, C. 2007. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies* 2(1):23–34.
- McDermott, D. (1978). Tarskian semantics, or no notation without denotation! *Cognitive Science* 2:277–282.
- Miles, A., and D. Brickley, eds. 2005. SKOS core guide. WorldWideWeb consortium. Latest version. <http://www.w3.org/TR/swbp-skos-core-guide>.
- Pollock, J., and R. Hodgson. 2004. *Adaptive information: Improving business through semantic interoperability, grid computing, and enterprise integration*. Wiley-Interscience.
- Popov, B., A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. 2003. KIM – Semantic annotation platform. In Proceedings of 2nd International Semantic Web Conference (ISWC2003), Florida, 20–23 Oct 2003, LNAI Vol. 2870, 834–849. Berlin/Heidelberg: Springer.
- Rocha, C., D. Schwabe, M.P., and de Aragao. 2004. A hybrid approach for searching in the semantic web. WWW 2004, New York, 17–22 May 2004.
- Salton, G., A. Wong, and C.S. Yang. 1997. A vector space model for automatic indexing. In *Readings in information retrieval*, eds. K. Sparck-Jones, and P. Willett. San Fransisco, CA: Morgan-Kaufman.
- Smith, B. 2003. Ontology. In *Blackwell guide to the philosophy of computing and information*, ed. L. Floridi, 155–166. Oxford: Blackwell.
- Smith, B., and C. Welty. 2001. Ontology: Towards a new synthesis. In Proceedings of the FOIS'01, Ogunquit, ME.
- Sparck-Jones, K., and P. Willett. 1997. *Readings in information retrieval*. San Fransisco, CA: Morgan-Kaufman.
- Spark Jones, K. 2004. What's new about the semantic web? Some questions. *SIGIR Forum* 38(2). <http://www.sigir.org/forum/2004D-TOC.html>. Accessed Dec 2004.
- Specia, L., and E. Motta. 2007. *Integrating folksonomies with the semantic web*. In Proceedings of the European Semantic Web Conference 2007 (ESWC 2007), Innsbruck, Austria. Heidelberg: Springer.
- Terziev, I., A. Kiryakov, and D. Manov. 2004. D1.8.1. Base upper-level ontology (BULO) guidance, report EU-IST integrated project (IP) IST-2003-506826 (SEKT). http://proton.semanticweb.org/D1_8_1.pdf.
- Vallet, D., M. Fernandez, and P. Castells. 2005. An ontology-based information retrieval model. In Proceedings of the 2nd European Semantic Web Conference, ESWC2005, Heraklion, Crete, May/June 2005, LNCS 3532/2005, eds. A. Gómez-Pérez, and J. Euzenat. Berlin: Springer.
- van Damme, C., M. Hepp, and K. Siorpaes. 2007. FolksOntology: An integrated approach for turning folksonomies into ontologies. Bridging the Gap Between Semantic Web and Web 2.0 Workshop, 4th European Semantic Web Conference, Innsbruck, Austria, June 2007.
- van Damme, C., T. Cornen, and E. Vandijck. 2008. Turning a corporate folksonomy into a lightweight corporate ontology. 11th International Business Information Systems Conference, BIS 2008, Innsbruck, Austria, May 2008. Heidelberg: Springer.
- van Ossenbruggen, J., L. Hardman, and L. Rutledge. 2002. Hypermedia and the semantic web: A research agenda. *Journal of Digital Information* 3(1), May 2002.

- Vitvar, T., M. Zaremba, M. Moran, and D. Fensel. 2007. SESA: Emerging technology for service-centric environments. *IEEE Software* 24(6):56–67, Nov/Dec 2007.
- Voorhees, E. 1998. Using WordNet for text retrieval. In *WordNet: An electronic lexical database*, ed. C. Fellbaum. Cambridge, MA: MIT Press.
- Warren, P., and J. Davies. 2007. Managing the risks from information through semantic information management. *BT Technology Journal* 25(1):178–191, Jan 2007.
- W3C Member Submission. 2004. OWL-S: Semantic markup for web services. <http://www.w3.org/Submission/OWL-S/>.

Chapter 10

WordNet

Christiane Fellbaum

10.1 Introduction

WordNet is a large electronic lexical database for English (Miller, 1995; Fellbaum, 1998a). It originated in 1986 at Princeton University where it continues to be developed and maintained. George A. Miller, a psycholinguist, was inspired by experiments in Artificial Intelligence that tried to understand human semantic memory (e.g., Collins and Quillian, 1969). Given the fact that speakers possess knowledge about tens of thousands of words and the concepts expressed by these words, it seemed reasonable to assume efficient and economic storage and access mechanisms for words and concepts. The Collins and Quillian model proposed a hierarchical structure of concepts, where more specific concepts inherit information from their superordinate, more general concepts; only knowledge particular to more specific concepts needs to be stored with such concepts. Thus, it took subjects longer to confirm a statement like “canaries have feathers” than the statement “birds have feathers” since, presumably, the property “has-feathers” is stored with the concept *bird* and not redundantly with the concept for each kind of bird.

While such theories seemed to be confirmed by experimental evidence based on a limited number of concepts only, Miller and his team were asking whether the bulk of the lexicalized concepts of a language could be represented with hierarchical relations in a network-like structure. The result was WordNet, a large, manually constructed semantic network where words that are similar in meaning are inter-related. While WordNet no longer aims to model human semantic organization, it has become a major tool for Natural Language Processing and spawned research in lexical semantics and ontology.¹

C. Fellbaum (✉)

Department of Computer Science, Princeton University, Princeton, NJ 08540, USA
e-mail: fellbaum@princeton.edu

¹For critical reviews of WordNet, see Kilgarriff (2000) and Lin (1999).

10.2 Design and Contents

WordNet is a large semantic network – a graph – in which words are interconnected by means of labeled arcs that represent meaning relations. Lexical relations connect single words while semantic-conceptual relations links concepts that may be expressed by more than one word.

Synonymy is the many-to-one mapping of word forms and concepts. For example, both the strings *boot* and *trunk* can refer to the same concept (the luggage compartment of a car). Under these readings, the two word forms are synonyms. WordNet groups synonyms into unordered sets, called synsets. Substitution of a synset member by another does not change the truth value of the context, though one synonym may be stylistically more felicitous than another in some contexts.

A synset lexically expresses a concept. Examples of synsets – marked here by curly brackets – are mail, post, hit, strike, and small, little. WordNet’s synsets further contain a brief definition, or “gloss,” paraphrasing the meaning of the synset, and most synsets include one or more sentences illustrating the synonyms’ usage. A domain label (“sports,” “medicine,” “biology,” etc.) marks many synsets.

Polysemy is the many-to-one mapping of meanings to word forms. Thus, *trunk* may refer to a part of a car, a tree trunk, a torso, or an elephant’s proboscis. In WordNet, membership of a word in multiple synsets reflects that word’s polysemy, or multiplicity of meaning. *Trunk* therefore appears in several different synsets, each with its own synonyms. Similarly, the polysemous word form *boot* appear in several synsets, once together with *trunk*, another time as a synonym of *iron boot* and *iron heel*, etc.

Synsets are the nodes or building blocks of WordNet. As a result of the interconnection of synsets via meaning-based relations, a network structure arises.

10.3 Coverage

WordNet in fact consists of four separate parts, each containing synsets with words from the major syntactic categories: nouns, verbs, adjectives, and adverbs. The current version of WordNet (3.0) contains over 117,000 synsets, comprising over 81,000 noun synsets, 13,600 verb synsets, 19,000 adjective synsets, and 3,600 adverb synsets. The separation of words and synsets for different parts of speech follows from the nature of the word class-specific semantic and lexical relations.

10.4 Relations

Besides synonymy, WordNet encodes another lexical (word-word) relation, antonymy (or, more generally, semantic contrast or opposition). Antonymy is psychologically salient, particularly among adjective pairs like *wet-dry* and *long-short*, but it is also encoded for verb pairs like *rise-fall* and *come-go*. (WordNet does not make the kind of subtle distinctions among the different kinds of semantic opposition drawn in, e.g., Cruse (1986)).

Another kind of lexical relation, dubbed “morphosemantic,” is the only one that links words from all four parts of speech. It connects words that are both morphologically and semantically related (Fellbaum and Miller, 2003). For example, the semantically related senses of *interrogation*, *interrogator*, *interrogate*, and *interrogative* are interlinked.

All other relations in WordNet are conceptual-semantic relations and connect not just words (synset members) but entire synsets. For each part of speech, different relations were identified.

10.5 Nouns in WordNet

Nouns comprise the bulk of the English lexicon, and the noun component of WordNet reflects this. Nouns are relatively easy to organize into a semantic network; WordNet largely follows the Aristotelian model of categorization.

10.5.1 Hyponymy

Noun synsets are primarily interconnected by the hyponymy relation (or hyperonymy, or subsumption, or the ISA relation), which links specific concepts to more general ones. For example, the synset gym shoe, sneaker, tennis shoe is a hyponym, or subordinate, of shoe, which in turn is a hyponym of footwear, footgear, etc. And gym shoe, sneaker, tennis shoe is a hypernym, or superordinate, of plimsoll, which denotes a specific type of sneaker. The relation is bi-directional; therefore, these examples express both that gym shoe, sneaker, tennis shoe is a type of footwear, footgear and that the category footwear, footgear is comprised of gym shoe, sneaker, tennis shoe (as well as other types of footwear, such as boot and over-shoe). Hyponymy is transitive, so plimsoll, by virtue of being a type of gym shoe, sneaker, tennis shoe, which is a type of footwear, footgear, is also a type of footwear, footgear.

Hyponymy builds hierarchical “trees” with increasingly specific “leaf” concepts growing from an abstract “root.” Noun hierarchies can be deep and comprise as many as fifteen layers, particularly for biological categories, where WordNet includes both expert and folk terms.

All noun synsets ultimately descend from a single root, *entity*. The next layer comprises three synsets: *physical entity*, *abstract entity*, and *thing*. Below these, we find the synsets *object*, *living thing*, *causal agent*, *matter*, *physical process*, *substance*, *psychological feature*, *attribute*, *group*, *relation*, *communication*, *measure*, *quantity*, *amount*, and *otherworld*.

The selection of these very broad categories was of course somewhat subjective and has engendered discussion with ontologists. On an empirical level, it remains to be seen whether wordnets for other languages draw the same fundamental distinctions.²

²We will refer to the Princeton WordNet as “WordNet” and databases in other languages as “wordnets,” indicating the fact that the proper name *WordNet* has become a common noun.

10.5.2 *Types vs. Instances*

Within the noun hierarchies, WordNet distinguishes two kinds of hyponymys, types and instances. Common nouns are types: *city* is a type of *location*, and *university* is a type of *educational establishment*. However, *New York* and *Princeton* are not types, but instances of *city* and *educational establishment*, respectively. Proper names are instances, and instances are always leaf nodes that have no hyponyms (Miller and Hristea, 2004).

While the Princeton WordNet does not distinguish roles from types and instances, some later wordnets do, e.g., EuroWordNet (Vossen, 1998). Thus, nouns like *pet* and *laundry* are encoded as types of *animal* and *garment*, respectively, on par with *poodle* and *robe*. This treatment does not satisfactorily reflect the categorial status of such nouns; on the other hand, it is doubtful whether a consistent labeling of role nouns is possible (David Israel, personal communication).

10.5.3 *Meronymy*

Another major relation among noun synsets is meronymy (or part-whole relation). It links synsets denoting parts, components, or members to synsets denoting the whole. Thus, *toe* is a meronym of *foot*, which in turn is a meronym of *leg* and so on. Like hyponymy, meronymy is bi-directional. WordNet tells us that a *foot* has *toes* and that *toe* is a part of a *foot*. Hyponyms inherit the meronyms of their superordinates: If a *car* has *wheels*, then kinds of *cars* (*convertible*, *SUV*, etc.) also have *wheels*. (But note that statements like “a toenail is a part of a leg,” though true, sound odd.)

Meronymy in WordNet actually encompasses three semantically distinct part-whole relations. One holds among proper parts or components, such as *feather* and *wing*, which are parts of *bird*. Another links substances that are constituents of other substances: *oxygen* is a constituent part of *water* and *air*. Members like *tree* and *student* are parts of groups like *forest* and *class*, respectively. Many more subtle kinds of meronymy could be distinguished (Chaffin, 1992).

10.6 Verbs

Verbs are fundamentally different from nouns in that they encode events and states that involve participants (expressed by nouns) and in that they have temporal extensions. The classic Aristotelian relations that work well to construct a network of noun synsets are not suitable for connecting verbs. Verb synsets are organized by several lexical entailment relations (Fellbaum, 1998b). The most frequently encoded relation is “troponymy”, which relates synset pairs such that one expresses a particular manner of the other. For example, *mumble* is a troponym of *talk*, and *scribble* is a troponym of *write*. Like hyponymy, troponymy builds hierarchies with several levels of specificity, but verb hierarchies are more shallow than noun hierarchies and rarely exceed four levels.

The particular manner encoded by troponyms is not specified, and troponymy is in fact a highly polysemous relation whose semantics are domain-dependent. For communication verbs, the medium distinguishes broad classes of verb hierarchies (*speaking, writing, gesturing*); motion verbs tend to be differentiated by such components as speed (*walking vs. running vs. ambling*).

Another relation among verb synsets is backward entailment, where the event encoded in one synset necessarily entails a prior event that is expressed by the second synset. Examples are *divorce* and *marry* and *untie* and *tie*. While the events in such pairs do not temporally overlap, those linked via a presupposition relation do. Examples are *buy* and *pay*: If someone buys something, he necessarily pays for it, and paying is a necessary part of the buying event. Finally, WordNet encodes a cause relation, as between *show* and *see* and *raise* and *rise*. Note that these relations are unidirectional.

A particular kind of polysemy is found in “auto-relations,” where a word form has a sense that expresses both the general and the specific concept, as in *drink*, *imbibe* and *drink*, *booze* (Fellbaum, 2000).

10.7 Adjectives

Antonymy is the prevailing relation among adjectives. Most adjectives are organized into “direct” antonym pairs, such as *wet-dry* and *long-short*.

Each member of a direct antonym pair is associated with a number of “semantically similar” adjectives, either near-synonyms or different values of a given scalar property. Thus, *damp* and *drenched* are semantically similar to *wet*, while *arid* and *parched* are similar to *dry*. These semantically similar adjectives are said to be “indirect” antonyms of the direct antonym of their central members, i.e., *drenched* is an indirect antonym of *dry* and *arid* is an indirect antonym of *wet* (Miller, 1998). For experimental work examining this theory see Gross et al., (1989).

WordNet also contains “relational” adjectives, which are morphologically derived from, and linked to, nouns in WordNet. An example is *atomic*, *nuclear*, which is linked to *atom*, *nucleus*.

10.8 Where do Relations Come from?

People often ask how the WordNet relations and the specific encodings were arrived at. Some of the relations, like hyponymy and meronymy, have been known since Aristotle. They are also implicitly present in traditional lexicographic definitions; a noun is typically defined in terms of its superordinated and the particular differentiae, or in terms of the whole entity of which the noun denotes a part. Verbs, too, are often defined following the classical genus-differentiae form.

Word association norms compile the responses people give to a lexical stimulus. Frequent responses are words that denote subordinate and superordinate concepts, or words that are semantically opposed to the stimulus words.

For adjectives, the responses are strikingly uniform and robust; thus, most people say *cold* when asked to respond with the word that comes to mind first when they hear *hot*. These data inspired the organization of adjectives in terms of antonymy (Miller, 1998).

To encode these relations for specific words and synsets, the WordNet team relied on existing lexicographic sources as well as on introspection. In addition, Cruse (1986) lists some test for synonymy and hyponymy. For example, the pattern “Xs and other Ys” identifies X as a hyponym (subordinate) of Y, rather than a synonym.

When the bulk of WordNet was compiled, large corpora were not yet available that could have provided a different aspect on semantic similarity: co-occurrence in identical or similar contexts. More recent lexicons are often constructed semi-automatically, relying heavily on the distributional patterns of word forms as a measure of similarity.

10.9 WordNet as a Thesaurus

Traditional paper dictionaries are necessarily organized orthographically so as to enable look-up. But this means that words that are semantically related are not found together, and a user trying to understand the meanings of words in terms of related words or words in the definition of the target word, must flip many pages.

By contrast, WordNet’s semantics-based structure allows targeted look-up for meaning-related words and concepts from multiple access points. But unlike in a traditional thesaurus such as Roget’s, the arcs among WordNet’s words and synsets express a finite number of well-defined and labeled relations.

10.10 Semantic Distance and Lexical Gaps

The WordNet relations outlined here sufficed to interrelate the words of English; this was not at all obvious from the start. But WordNet’s apparently simple structure hides some unevenness. First, the meaning difference, or semantic distance, between parent and child nodes varies. For example, while verbs like *whisper*, *mumble*, and *shout* all seem equidistant from their parent *talk*, the distance between *talk* and its direct superordinate, *communicate*, seems much larger. This can be seen in the fact that *whisper*, *mumble* and *shout* can be fairly easily replaced by *talk* in many contexts without too much loss of information, whereas the substitution of *talk* with *communicate* would be very odd in many contexts.

A question related to semantic distance concerns lexical gaps, arguably concepts that for no principled reason are not linguistically labeled. For example, the lexicon suggests that nouns like *car*, *bicycle*, *bus*, and *sled* are all direct subordinates of *vehicle*. But this group of “children” seems heterogeneous: *sled* stands out for several reasons, in particular for not having wheels. To draw what appears like a

major distinction among the many vehicles, WordNet introduced a synset *wheeled vehicle*. The argument is that people distinguish between the category of wheeled vehicles and vehicles moving on runners independently of whether this distinction is lexically encoded in their language. One would expect other languages to label these concepts and show that the lack of an English word is purely accidental. (In fact, German has a word, *Kufenfahrzeug*, for *vehicle on runners*).³

Adding nodes in places where lexical gaps are perceived reduces the semantic distance among lexicalized categories but presents a more regular picture of the lexicalization patterns than warranted by purely linguistic data. Thus, the introduction of lexical gaps is a matter of discussion among wordnet builders. On the other hand, it is common practice in ontology, where it is usually assumed concepts are independent of natural language labels.

10.11 WordNet as an Ontology

Because of its rigid structure, WordNet is often referred to as an ontology; indeed, some philosophers working on ontology have examined WordNet's upper structure and commented on it. For example, Gangemi et al. (2002a, b) and Oltramari et al. (2002) have made specific suggestions for making WordNet more consistent with ontological principles. But the creators of WordNet prefer to call it a "lexical ontology," because its contents – with few exceptions – are concepts that are linguistically encoded and its structure is largely driven by the the lexicon. By contrast, many ontologists emphasize that an ontology is language-independent and merely uses language to refer to concepts and relations. Ontologies are usually understood to be knowledge structures rather than lexicons. For further discussion on the lexicon-ontology difference see Pease and Fellbaum (2009).

10.12 WordNet and Formal Ontology

WordNet has been linked to formal ontologies (Gangemi et al., 2002a; Niles and Pease, 2003). Concepts in one ontology, SUMO (Suggested Upper Merged Ontology, Niles and Pease, 2001; Niles and Pease, 2003; Chapter 11, Controlled English to Logic Translation, Pease and Li, this volume) have been linked to synsets not only in the Princeton WordNet but to many wordnets in other languages as well.

SUMO is a formal ontology stated in a first-order logic language called SUO-KIF. SUMO contains some 1,000 terms and 4,000 axioms using those terms in SUO-KIF statements. These axioms include some 750 rules. SUMO is an upper ontology, covering very general notions in common-sense reality, such as time, spatial relations, physical objects, events and processes.

³Fellbaum and Kegl (1989) argue for lexical gaps in the verb lexicon on syntactic grounds.

A mid-level ontology (MILO) was created to extend SUMO with several thousand more terms and associated definitions for concepts that are more specific. In addition, domain ontologies cover over a dozen areas including world government, finance and economics, and biological viruses. Together with SUMO and MILO they include some 20,000 terms and 60,000 axioms.

Niles and Pease (2003) manually mapped the formally defined terms in SUMO to synsets in WordNet. Three types of mappings were made: rough equivalence, subsuming, and instance. In addition, mappings were made for senses that appeared to occur frequently in language use, based on the SemCor semantic concordance (Miller et al., 1993). New concepts were created in the MILO as needed and linked to the appropriate synsets. SUMO, MILO, and the domain ontologies have been linked to wordnets in several other languages as well (for details on the linking see Pease and Fellbaum, 2009).

For example, the synset *artificial satellite, orbiter, satellite* maps to the formally defined term of *ArtificialSatellite* in SUMO. The mapping is an “equivalence” mapping since there is nothing that appears to differentiate the linguistic notion from the formal term in this case. A more common case of mapping is a “subsuming” mapping. For example *elk* maps to the SUMO term *HoofedMammal*. WordNet is considerably larger than SUMO and so many synsets map to the same more general formal term. As an example of an “instance” link, the synset *george washington, president washington, washington* is linked to the SUMO term *Human*. Because WordNet discriminates among different senses of the same linguistic token, the synset *evergreen state, wa, washington* is linked via an “instance” relation to the term *StateOrProvince*.

10.13 Wordnets in Other Languages

Since the 1990s, wordnets are being built in other languages. The first, EuroWordNet (EWN, Vossen, 1998), encompasses eight languages, including non-Indo-European languages like Estonian and Turkish. EuroWordNet introduced some fundamental design changes that have been adopted by many subsequent wordnets. Crucially, all wordnets are linked to the Princeton WordNet.

10.14 The EuroWordNet Model

Wordnets were constructed for each language following one of two strategies. The first, dubbed “Expand,” was to translate the synsets of the Princeton WordNet into the target language, making adjustments as needed (see below). The second, dubbed “Merge,” was to develop a semantic network in the target language from scratch and subsequently link it to the Princeton WordNet.

Several innovations were introduced. In contrast to Princeton WordNet’s strict limitation to paradigmatic relations, the wordnets built for EWN encode many cross-POS links. For example, syntagmatically associated nouns and verbs, such as the

pair *student* and *learn* are linked. Another innovation are conjunctive and disjunctive relations. Conjunctive relations allow a synset to have multiple superordinates. Thus *knife* is both a kind of *eating utensil* and a kind of *weapon*. Another example is *albino*, which can be a kind of *person*, *animal*, or *plant*. Double parenthood can capture the Type vs. Role distinction discussed earlier.

An example for a disjunctive relation is *airplane* and its meronyms *propellor* and *jet*; a given type of airplane has either, but not both, parts (Vossen, 1998). The possibility of disjoint parts can reduce the proliferation of artificial nodes, such as *propeller plane* and *jet plane*.

Each wordnet that is part of EuroWordNet relates to three language-neutral components: the Top Concept Ontology, the Domain Ontology, and Interlingual Lexical Index (ILI).

The Top Concept Ontology is a hierarchically organized set of some 1,000 language-independent core concepts that are expressed in all wordnets. The Domain Ontology consists of a set of topical concepts like “medicine” and “traffic”; unlike the unstructured list of domain labels in the Princeton WordNet, the domain concepts form a hierarchy.

In contrast to the individual wordnets, which are semantic networks with hierarchical relations, the ILI is an unstructured, flat list of lexicalized concepts. Each is represented by a synset and an English definition of the concept expressed by the synset members. The ILI started out as the Princeton WordNet, with each synset being assigned a unique identification number, or “record.” The words and synsets of the languages of EWN were mapped, to the extent possible, onto the synsets in the ILI, and the record identification number was attached to the corresponding word or synset in the target language.

In those cases where a language has one or more words expressing a concept that is not lexicalized in English (i.e., lacking corresponding English words), a new record was created in the ILI with just an identification number but without English lexemes; this record includes a pointer to the synset in the source language. In this way, the ILI came to include, besides WordNet’s synsets, records for all concepts that are lexicalized in one or more EuroWordNet language but not in English. The ILI thus constitutes the superset of all concepts included in all European wordnets.

By means of the records, the ILI mediates among the synsets of the individual languages. Equivalent concepts and words across languages can be determined by referencing the appropriate ILI records.

Maintaining the ILI is a flat list of entries and restricting the encoding of lexical and semantic relations to each of the language-specific wordnets avoids the problem of crosslinguistic mismatches in the patterns of lexicalization and hierarchical organization. For example, Vossen (2004) cites the case of English *container*, which has no counterpart in Dutch. Dutch does have, however, words for specific kinds of containers, like *box*, *bag*, etc. If the ILI had taken over the English WordNet’s hierarchical structure, where *container* is a superordinate of *box*, *bag*, etc., mapping to Dutch would be problematic. Instead, the Dutch wordnet simply maps the Dutch words for *box*, *bag*, etc., to its (Dutch) lexicalized superordinate (implement) and disregards the *container* level.

As the number of wordnets grows, the need for connecting them to one another becomes pressing; moreover, linking should be able not only accurate matching of cross-linguistically encoded concepts but also allow for the expression of meanings that are specific to a language or culture. Fellbaum and Vossen (2007) and Vossen and Fellbaum (2009) propose the creation of a suitable infrastructure dubbed the “Global Grid.”

Interconnected multilingual wordnets carry tremendous potential for crosslinguistic NLP applications and the study of universal and language-specific lexicalization patterns.

10.15 Global WordNets

Currently, wordnet databases exist for several dozen languages (see Singh, 2002; Sojka et al., 2004, 2006) and new ones are being developed.⁴ Virtually all wordnet developments follow the methodology of EuroWordNet described earlier. But wordnets for typologically distinct languages pose novel challenges especially with respect to the notions “concept” and “word,” which must be defined to determine synsets and synset membership. One challenge is the morphology of agglutinative languages like Turkish, Estonian, Tamil and Basque (Bilgin et al., 2004; Kahusk and Vider, 2002; Thiyagarajan 2002; Agirre et al., 2002), where multiple affixes that carry grammatical and lexical meaning are added to a stem to form a single long “word”. For example, does a diminutive formed via an affix express a concept distinct from the base form or are they merely lexical variants? Diminutives are arguably independent words, and they could be included in a wordnet as such, with a pointer expressing a “diminutive” relation to the base form.

Even more challenging are languages like Hebrew and Arabic, where words are generated from a triconsonantal root that constitutes a kind of “super-concept” but that does not have lexical status itself; words whose meanings share the core meaning of the root are derived from it via the addition of vowels (Black et al., 2006).

For Chinese, Wong and Pala (2004) propose to exploit the semantics inherent in the Chinese writing system. A character typically consists of two radicals, one of which carries meaning while the other indicates the pronunciation. Characters and the concepts they express can be grouped and related to one another based on the meaning-carrying radical, at least at the top and middle level of the hierarchies.

10.16 WordNet as a Tool for Natural Language Processing

WordNet’s design and electronic format have proved useful for a wide range of Natural Language Processing (NLP) applications, including mono- and crosslinguistic information retrieval, question-answering systems, and machine translation.

⁴See the website of the Global WordNet Organization, <http://www.globalwordnet.org>

All these tasks face the challenge of word sense identification posed by lexical polysemy. Statistical approaches can identify the context-intended sense in many cases but are limited. WordNet facilitates alternative or complementary symbolic approaches to word sense discrimination, as it allows automatic systems to detect and measure the semantic relatedness of polysemous words that co-occur in a context.⁵

10.17 Conclusions

WordNet represents a new approach – made possible by its electronic format – towards revealing the systematic ways in which a language maps concepts onto words. WordNet deliberately focuses on the lexicon, but its rigid structure and representation of upper-level words and concepts have sometimes invited its comparison to an ontology, a language-independent knowledge structure. Mapping concepts in formal ontologies to synsets in wordnets maintains that distinction and sheds light on concept-word mapping patterns.

Crosslinguistic wordnets show significant overlap at the top levels but diverge on the middle and lower levels, often due to language-specific lexicalization patterns. Further research on WordNet and the development of wordnets in genetically unrelated and typologically diverse languages should advance our understanding of universal and language-specific conceptual and lexical structure.

Acknowledgements Work partially supported by contracts from The Disruptive Technology Office, REFLEX, and the U.S. National Science Foundation.

References

- Agirre E., O. Ansa, X. Arregi, J.M. Arriola, A. Diaz de Ilaraza, E. Pociello, and L. Uria. 2002. Methodological issues in the building of the Basque Word-Net: Quantitative and qualitative analysis. In: Proceedings of the First International WordNet Conference, Mysore, India: CIIL.
- Bilgin, O., O. Cetimo, and K. Oflaver. 2004. Building a word-net for Turkish. *Romanian Journal of Information Science and Technology* 7:1–2, 163–172.
- Black, W., S. Elkateb, H. Rodriguez, M. Alkhalifa, P. Vossen, A. Pease, M. Bertran, and C. Fellbaum. 2006. The Arabic WordNet project. *Proceedings of the Conference on Lexical Resources in the European Community*, Genoa, Italy.
- Baker, C., C.J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet project. *Proceedings of the COLING-ACL Montreal*, Canada.
- Chaffin, R. 1992. The concept of a semantic relation. In *Frames, fields, and contrasts: New essays in semantic and lexical organization*, eds. A. Lehrer and E.F. Kittay, 253–288. Hillsdale, NJ: Lawrence Erlbaum.
- Collins, A. M., and M. R. Quillian. 1969. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8:240–247.
- Cruse, A. 1986. *Lexical semantics*. Cambridge: Cambridge University Press.
- Fellbaum, C. ed. 1998a. *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.
- Fellbaum, C. 1998b. A semantic network of english verbs. In ed. C. Fellbaum, 69–104.

⁵For a bibliography on WordNet-based work see <http://lit.csci.unt.edu/wordnet>.

- Fellbaum, C. 2000. Autotroponymy. In *Polysemy*, eds. Y. Ravin, and C. Leacock, 52–67. Cambridge, MA: Cambridge University Press.
- Fellbaum, C., and J. Kegl. 1989. Taxonomic structure and object deletion in the english verbal system. In Proceedings of the 6th Eastern States Conference on Linguistics, eds. K. deJong, and Y. No, 94–103. Columbus, OH: Ohio State University.
- Fellbaum, C., and G. A. Miller. 2003. Morphosemantic links in Word-Net. *Traitement Automatique des Langues* 44(2):69–80.
- Fellbaum, C. and P. Vossen. 2007. Connecting the universal to the specific: Towards the global grid. In Proceedings of the First International Workshop on Intercultural Communication, Kyoto University, Japan. Reprinted in Intercultural Collaboration: First International Workshop. LNCS 4568, eds. T. Ishida, S.R. Fussell, and P.T. J. M. Vossen, 1–16. New York: Springer.
- Gangemi, A., N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. 2002a. *Sweetening ontologies with DOLCE*. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), 166, LNCS 2473.
- Gangemi, A., N. Guarino, A. Oltramari, and S. Borgo. 2002b. Cleaning-up WordNet's top-level. In Proceedings of the First International WordNet Conference, Mysore, India: CIIL.
- Gross, D., U. Fischer, and G.A. Miller. 1989. The organization of adjectival meanings. *Journal of Memory and Language* 28:92–106.
- Kahusk, N., and K. Vider. 2002. Estonian WordNet benefits from word sense disambiguation. In Proceedings of the First Global WordNet Conference, 26–31. Mysore, India: Central Institute of Indian Languages. .
- Kilgarriff, A. 2000. Review of WordNet. *Language* 76:706–708.
- Lin, D. 1999. Review of WordNet: An electronic lexical database. *Computational Linguistics* 25:2.
- Miller, G.A. 1995. WordNet: A lexical database for english. *Communications of the ACM* 38: 39–41.
- Miller, G.A. 1998. Nouns in WordNet. In *Word-Net: An electronic lexical database*, ed. C. Fellbaum, Cambridge, MA: MIT Press.
- Miller, G. A., and C. Fellbaum. 1991. *Semantic networks of english*. Cognition, special issue, eds. B. Levin, and S. Pinker, 197–229. Reprinted in *Lexical and Conceptual Semantics*, eds. B. Levin, and S. Pinker, 197–229. Cambridge, MA: Blackwell.
- Miller, G.A., and F. Hristea. 2004. WordNet nouns: Classes and instances. *Computational Linguistics* 32:1.
- Miller G., C. Leacock, T. Randee, R. Bunker. 1993. A semantic concordance. In Proceedings of the 3rd DARPA Workshop on Human Language Technology, Plainsboro, NJ, 303–308.
- Miller, K. 1998. Modifiers in WordNet. In *Word-net: An electronic lexical database*. ed. C. Fellbaum, Cambridge, MA: MIT Press.
- Niles, I., and A. Pease. 2001. Towards a standard upper ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, Ogunquit, Maine.
- Niles, I., and A. Pease. 2003. Linking lexicons and ontologies: Mapping WordNet to the suggested upper merged ontology. In Proceedings of the IEEE International Conference on Information and Knowledge Engineering, 412–416.
- Niles, I. and A. Terry. 2004. The MILO: A general-purpose, mid-level ontology. In Proceedings of the International Conference on Information and Knowledge Engineering, Las Vegas, Nevada.
- Oltramari, A., A. Gangemi, N. Guarino, and C. Masolo. 2002. Restructuring WordNet's top-level: The ontoclean approach. In Proceedings of LREC, Las Palmas, Spain.
- Pease, A. 2003. The sigma ontology development environment. Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems. Proceedings of CEUR 71.
- Pease, A., and C. Fellbaum. 2009. Formal ontology as interlingua: The SUMO and WordNet Linking Project. In *Ontologies and the Lexicon*, eds. C.-R. Huang, N. Calzolari, A. Gangemi, A. Lenci, A. Oltramari, and L. Prévot, 29–43. Cambridge Studies in Natural Language Processing. Cambridge: Cambridge University Press.
- Sojka, P., K. Pala, P. Smrz, C. Fellbaum, and P. Vossen, eds. 2004. Proceedings of the Second International WordNet Conference. Brno, Czech Republic: Masaryk University.

- Sojka, P., K.-S. Choi, C. Fellbaum, and P. Vossen. 2006. *Proceedings of the third global WordNet conference*. Brno, Czech Republic: Masaryk University.
- Thiyagarajan, S., S. Arulmozi, S. Rajendran. 2002. Tamil WordNet. In *Proceedings of the First Global WordNet Conference*, Mysore, India: CIIL.
- Vossen, P. 1998. ed. *EuroWordNet*. Dordrecht: Kluwer.
- Vossen, P. 2004. EuroWordNet: A multilingual database of autonomous and language-specific wordnets connected via an interlingual index. *International Journal of Lexicography* 17(2):161–173.
- Vossen, P., and C. Fellbaum. 2009. Universals and idiosyncracies in multilingual wordnets. In *Multilingual lexical resource*, ed. H. Boas, Hans, 319–345. Berlin: de Gruyter.
- Wong, S.H.S., and K. Pala. 2004. Chinese characters and top ontology in EuroWordNet. In *Proceedings of the First Global WordNet Conference*, ed. U.N. Singh, 224–233. Mysore, India: CIIL.

Chapter 11

Controlled English to Logic Translation

Adam Pease and John Li

11.1 Introduction

It has long been a goal in computer science for users to communicate with computers in human language. Instead of complex menu systems or command languages, people would certainly rather interact with a computer in the same way one would interact with other people. While there have been many advances in creating computer software that handles English, or other languages, understanding is largely limited to search and retrieval. That is, a user enters a phrase, keywords or a question, and the computer attempts to find the best match between the user's input, and some body of stored text. The computer doesn't understand the question, or the content that it is searching. It can't combine information across documents to create a novel answer to a novel question. It can't understand simple logical conclusions that any human could draw from the stored text. If natural language could be translated to logic, the computer would have a form that could be used for inference, and a degree of real understanding.

We have developed a program which takes a restricted form of natural language and automatically translates it to logic. The Controlled English to Logic (CELT) system (Pease and Murray, 2003) has a deep understanding of queries and statements. CELT performs syntactic and semantic analysis on English input, and transforms it to logic. CELT allows new information to be added to a knowledge base and be immediately available for queries.

There have been a small number of previous efforts that are similar to our approach, ACE (Fuchs, 1999) being the most notable, and is the work that inspired CELT. However, a key advance in our work is that the output of the language translation system uses terms in an existing large ontology, the Suggested Upper Merged Ontology (SUMO) (Niles and Pease, 2001). Previous efforts have lacked such a connection, and resulted in logical statements in which terms have meaning only to the extent they are used in a series of sentences from the user. By

A. Pease (✉)
Articulate Software, Mountain View, CA, USA
e-mail: apease@articulatesoftware.com

linking statements to SUMO, each term has a wealth of existing definition, much as people have a wealth of meaning and understanding behind each word that is used in communication. One might also question whether this work is truly new, since most of the issues in the semantics of English of concern to CELT have been presented in linguistics research over the years. The challenge is in bringing all this material together in one computational model, and using a single consistent ontology.

The Suggested Upper Merged Ontology (SUMO) is a freely available, formal ontology of about 1,000 terms and 4,000 definitional statements. It is provided in first order logic, and also translated into the OWL semantic web language. It is now in its 75th free version; having undergone 5 years of development, review by a community of hundreds of people, and application in expert reasoning and linguistics. SUMO has been subjected to formal verification with an automated theorem prover. SUMO has been extended with a number of domain ontologies, which are also public, that together number some 20,000 terms and 70,000 axioms. SUMO has also been mapped to the WordNet lexicon (Fellbaum, 1998; Chapter 10 by Fellbaum, this volume) of over 100,000 noun, verb, adjective and adverb word senses (Niles and Pease, 2003), which not only acts as a check on coverage and completeness, but also provides CELT's lexicon. SUMO and all the associated tools and products are made available at www.ontologyportal.org

Although CELT uses a simplified syntax it does not limit the parts of speech or the number of word senses a word can have, at least to the extent that by using WordNet, CELT's vocabulary grows as WordNet grows. More importantly, CELT is not a domain specific system as with (Allen et al., 1994). It is a completely general language, but one which can be specialized and extended for particular domains along with domain specific vocabulary.

We see an analogy with the PalmPilot and our work. The Apple Newton was an innovative product that attempted to do recognition of unrestricted handwriting, after some training. The Newton was a failure. The problem was simply too hard to be tractable at the time. The system did not correctly interpret handwriting enough of the time to be useful. The PalmPilot took a different approach. It requires handwriting to be one character at a time, in a special alphabet. These restrictions eliminate most of the hard problems in handwriting recognition. A small burden is placed on the user, and in return the user is provided a very useful product. People will change their behavior if the change is relatively small in proportion to the benefit derived. Placing the jobs that are hard for machines and easy for people in the domain of the human user can make an impossible job practical. People have been predicting the arrival of full text understanding ever since the beginning of AI. The realization of this prediction is usually a constant 10 years from the time when the prediction is being made. The time horizon keeps being extended and is unlikely to arrive soon. The best solution may be to simplify the problem in a smart way.

CELT uses a parsing approach that relies on a controlled English input. This means that the user asks queries in a specified grammatical format. This subset of English grammar is still quite extensive and expressive. The advantage of the controlled English is that when the grammar and interpretation rules are restricted,

then every sentence in the grammar has a unique interpretation. This eliminates the problems of ambiguity with other approaches that would result in retrieving non-appropriate answers.

Another way to look at language research is that most work has focused on handling all language at a very shallow level of understanding, and advances now work in the direction of gradually increasing the level of understanding while maintaining coverage of all possible utterances. CELT takes the opposite approach of starting from complete understanding, at least to the extent possible in formal logic, of a very restricted subset of English. Research on CELT has focused on increasing the range of understandable sentences, while maintaining the requirement to have complete understanding of the semantics of the sentences handled.

CELT's controlled English is meant to provide a syntax with a lexical semantics, which means that the sense of a sentence depends only upon the sentence, how the words are put together so as to select one out of many possible meanings for each, and not the context within which it is spoken or written, for example, a paragraph, document, conversation, or the gestalt of a perceived situation. This converts the English syntax into that of a formal as opposed to natural language. This approach appropriately partitions application tasks into components which would be hard to handle by machine (extracting the intended sense of a sentence) and hard to handle by a human (quickly and efficiently interpreting extensive text, performing semantically rich Internet searches, etc). The conversion from a fragment of a natural language (English) to a formal language (controlled English) puts the converted language fragment in the same class with the mathematical vehicle used for inferencing with the extracted text (formal logic).

11.2 WordNet Mappings

We have mapped all of the over 100,000 word senses in WordNet to SUMO, one at a time, by hand. Our original work is described in detail in a previous publication (Niles and Pease, 2003). Since the original version, we have modified the mappings to keep up to date with the latest versions of WordNet, which resulted in a port to WordNet 2.0 and then 3.0. We have also revised the mappings to point to more specific terms in the mid-level and domain ontologies that extend SUMO.

Briefly, WordNet is organized as a list of "synsets" or synonym sets. Synsets each have a dictionary definition and a set of links to other synsets. We assigned each synset a link to a particular SUMO term, or in a few cases, links to several terms. WordNet is much larger than SUMO so many synsets link to SUMO terms that are more general. For example "to flee", in the sense of "run away quickly" is linked to SUMO's **Running**, since there is no more specific term available. The word "valley" however has only one sense in WordNet and it is linked to the equivalent SUMO term of **Valley**. Note that these links are not based on the name of the SUMO term, but rather on the meanings of the formal term and the linguistic term. For example, the informal synset "seven seas" links to the SUMO term **Ocean**.

Having a lexicon and an ontology also enforces a certain discipline of clearly separating linguistic and ontological information. Many ontologies, especially those

defined in computer languages that are largely limited to taxonomic information, rely heavily on human intuition about natural language term names for their definition. SUMO term names are not strictly part of the definition of any term. Term meanings come solely from the formal axioms on the term. Term names could be replaced with arbitrary unique symbols and still have the same logically defined meanings. By keeping the lexicon separate from the ontology, each can follow principles of organization specific to their goals as independent works. Many lexicons for languages other than English have been created (Vossen and Fellbaum, 2002) and many of those have been mutually linked. As a result, SUMO has links to many languages in addition to English, further keeping clear the distinction between formal terms and their correspondence to human languages.

11.3 Simple Parsing and Interpretation

CELT first parses a sentence to determine the parts of speech for each word. It creates a parse “tree”, such as in Fig. 11.1, that groups these words. For the sentence “Mike reads the book.” there is a proper noun “Mike”, then a verb, determiner, another noun and then a period. The pair of words “the book” forms a noun phrase, noted as “NP”. The phrase “reads the book” is a verb phrase, or “VP”, and so on.

CELT uses the order of the words, and a dictionary that labels words as nouns, verbs and so forth, in order to determine the parts of speech. The dictionary used is WordNet, augmented with a list of common proper names.

Once CELT has determined the parts of speech of each word, it must also determine the particular sense of each word that is intended. For example, is “reads” the sense of reading text in a book, or the sense of to understand, as in “I read you loud and clear.” Only by determining the proper sense can CELT find the proper related SUMO term to translate to, which would then either be **Reading** or **Communication**, respectively. This is discussed further in Section 11.3.1 below.

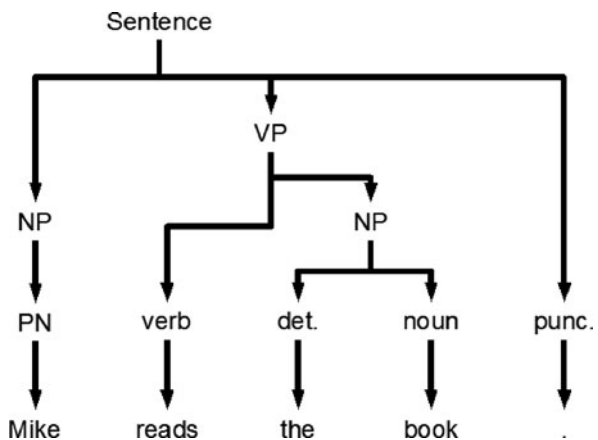


Fig. 11.1 Simple parse tree

Once the CELT parser has determined parts of speech and word senses, it can begin to bring each linguistic element together into a logical expression. SUMO takes a Davidson (1980) approach to describing events, in which verbs denoting actions are reified as instances, and the various elements of the event are related to the instance. In this simple example, the subject of the sentence, “Mike” is therefore the agent of the action, and so is brought together with an instance of a “read” event by the SUMO relation **agent**. The object of the sentence is related to the event with the SUMO relation **patient**. A different set of relationships are employed for different linguistic constructions, such as stative verbs, or quantified sentences, and several of these are described further in Section 11.4 below. The value of the resulting logical form is that it can be subjected to deductive inference with a standard first order theorem prover, and yielding new knowledge deduced from known facts and input from the CELT user. This can be contrasted with “statistical” based language understanding approaches that yield approximate and non-logical results for which deductions, if performed, are not truth-preserving.

To elaborate, the SUMO term reading has a number of logically specified truths that are consistent as long as one employs the standard semantics of first order logic (actually SUMO employs some higher-order logical expressions, but let us leave that detail for another paper). One can follow deduction even a hundred steps long and the result will not yield a contradiction. In contrast, although WordNet contains much valuable information, it simply is not intended nor able to support deductions of arbitrary length. For example, a specific sense of “to read” (synset 200626428) does entail the immediate parent (hypernym) synset of “verbalize” (synset 200941990 etc) and its parent “communicate” but that does not necessarily entail that one “interact(s)” (the next parent in turn, which is problematic since one can verbalize to one’s self, at least it appears so from the WordNet definition of that synset). While locally consistent, it is not a logical product. Further, nothing in a linguistic product such as WordNet allows one to conclude as does SUMO that **Reading** necessarily involves a **Text** (with a very specific, logical, and formally specified definition of what a **Text** is), that a **Text contains Information**, and any of the near infinite number of deductions that are possible in turn from each of those small conclusions. This discussion is not intended in any way to criticize WordNet, which is an essential part of CELT, but rather to emphasize the different functions that WordNet and SUMO support in CELT’s processing.

11.3.1 Word Sense Disambiguation

Our method for word sense disambiguation (WSD) is based on a large data set of English sentences that have been annotated by hand to specify the precise sense of each word. The data set used is the Brown Corpus (Kucera and Francis, 1967), and the markup of that corpus is called SemCor (Landis et al., 1998). This work is not new, and in fact is rather poor compared to the state of the art. Its only virtues are that the sources used are free, and the approach is simple. We plan on adopting a more sophisticated approach in the future. We began by processing SemCor to generate

a map in which each key is a particular word sense and the value is a list of words and frequencies that indicate how many times the given sense occurred in a sentence with each of a set of words. One weakness of SemCor is that it is not as large as many modern corpora used for this purpose, and key words that discriminate a particular sense may only occur a few times together with that sense in the Brown Corpus. There are also many senses in WordNet that do not occur at all in that corpus. We used two methods to improve the statistical significance of the entries. In support of our first method, many WordNet senses are very “fine-grained”, which is to say that some senses indicate very small differences in word sense that may not even be listed in most dictionaries. There are often cases where very similar word senses may map to the same SUMO concept. When that occurs, we have “collapsed” the sense entries, and added the co-occurrence data together. The second method used to improve performance over a simple use of SemCor relates to a recent effort by Princeton to disambiguate manually all the words in English definitions of senses in WordNet. We are processing these sentences just as with SemCor to add these statistics to our overall set. A key improvement resulting from this new work is that we will have at least some co-occurrence statistics on all the word senses in WordNet.

An additional method we have implemented is to use the entire history of a CELT dialog in WSD. In a very short sentence such as “Bob runs to the store.” it is very hard to get the sense of the highly polysemous word “run” even with a large manually disambiguated corpus. But, combined with previous sentences in a dialog such as “Bob is training for a marathon.” and “Bob bought new sneakers.” getting the correct result is much more likely.

11.4 Issues in Translation

11.4.1 Case Roles and Word Order

In English, word order is Subject-Verb-Object (except in the case of passive voice, which CELT does not handle). The roles of Subject, Object (and, when present, Indirect Object) have a correspondence to SUMO’s set of CaseRole(s), which define different kinds of participation in events. Continuing with the preceding example, “Mike” is the subject and “book” is the direct object. In SUMO, the **agent** relation

```
(exists (?M ?B ?R)
  (and
    (instance ?R Reading)
    (agent ?R Mike-1)
    (instance Mike-1 Human)
    (attribute Mike-1 Male)
    (instance ?B Book)
    (patient ?R ?B)))
```

Fig. 11.2 Case role example

brings together the performer of an action, and the action itself. The **patient** relation specifies a participant in the action. The output from CELT also specifies the types of the participants (Fig. 11.2).

11.4.2 Statives

The type of verb in a sentence has a critical role in interpretation of meaning. Most verbs indicate occurrence of an event or process. “Mike reads the book.” indicates that a **Reading** event is taking place. Other verbs however connote relationships. “Mike owns the book.” indicates that there is the relationship of ownership between Mike and the book. There is no “owning” event that takes place. Such verbs are called “statives”.

To complicate things, some stative verbs have an appropriate non-Process interpretation. For example “Dickens writes Oliver Twist.” should generate (**authors Dickens OliverTwist**). However, once that stative is augmented with a time or a place, then we have two things that are being said, the timeless fact, and the event. For example “Dickens writes Oliver Twist in 1837.” makes both the statement of authorship above, and the additional statement about a **Process** occurring in a particular year (Fig. 11.3).

Fig. 11.3 Stative example

```
(and
  (authors Dickens OliverTwist)
  (exists (?EV)
    (and
      (instance ?EV Writing)
      (agent ?EV Dickens)
      (equals (YearFn 1837) (WhenFn ?EV))
      (result ?EV OliverTwist))))
```

11.4.3 Attributes

In the SUMO-WordNet mappings we map most nouns to classes. However, some nouns can be better mapped to attributes that also imply class membership. A noun mapped to a **Position**, should be a **Human** that has the indicated **Position** as an attribute. For example, given the mapping from “pianist” to **Musician**, the sentence “Bob is a pianist.” results in Fig. 11.4.

```
(and
  (attribute Bob-1 Male)
  (instance Bob-1 Human)
  (attribute Bob-1 Musician))
```

Fig. 11.4 Stative example

11.4.4 Counting

CELT handles simple expressions about singulars, plurals, mass nouns and countability. “Bob kills 5 rats.” generates Fig. 11.5.

Fig. 11.5 Counting example

```
(exists
  (?CLNrats ?event)
  (and
    (attribute Robert-1 Male)
    (forall
      (?I)
      (=>
        (member ?I ?CLNrats)
        (instance ?I Rat)))
    (instance Robert-1 Human)
    (instance ?CLNrats Collection)
    (member-count ?CLNrats 5)
    (agent ?event Robert-1)
    (instance ?event Killing)
    (patient ?event ?CLNrats)))
```

11.4.5 Copula Expressions

CELT handles the many different meanings of the verb “to be”, which is what linguists call the “copula”. “Bob is a pianist.” is relatively straightforward, as shown above in the section on attributes. However, copula expressions can also be general statements about classes, as in “An apple is a fruit.”, which generates a subclass expression as follows:

```
(subclass Apple Fruit)
```

CELT provides the same translation for “Apples are fruits.” since the two are semantically equivalent.

11.4.6 Prepositions

Prepositions can have significantly different interpretations such as in “Bob is on the boat.” (Fig. 11.6). In contrast, the sentence “The party is on Monday.” refers to

```
(exists (?boat)
  (and
    (attribute Robert-1 Male)
    (instance Robert-1 Human)
    (instance ?boat Watercraft)
    (orientation Robert-1 ?boat On)))
```

Fig. 11.6 Preposition example

Fig. 11.7 Another preposition example

```
(exists (?party ?monday)
  (and
    (instance ?party SocialParty)
    (instance ?monday Monday)
    (during ?party ?monday)))
```

Table 11.1 Prepositions, class membership and relations

Preposition	Class	SUMO relation
at, in, on	location	location
at, in, on	time	time
for	person	destination
for, through	time	duration
with	person	agent
with	object	instrument
across	path	traverses
within, into	object	properlyFills
from	object	origin
from	time	BeginFn
through	object	traverses
until	time	EndFn
after	time	greaterThan
before	time	lessThan

a time, rather than a location, and therefore is translated as in Fig. 11.7. A list of many of the different impacts of argument type on the translations of prepositions is given in Table 11.1.

In order to determine the type of elements in the sentence we use SUMO class membership as shown in Fig. 11.8.

CELT type	SUMO class
time	TimeMeasure, Process
person	Human, OccupationalRole, SocialRole
object	all others
mass	Substance, Food, and special words such as money/furniture/data/life/beauty/truth/crime/law/education
count	all others

Fig. 11.8 Finding CELT types with SUMO class membership

11.4.7 Quantification

Quantification statements are generally those which use words like “every”, “all” or “some”. For example, “Every boy likes fudge.” results in the output shown in Fig. 11.9, and “Some horses eat hay.” results in Fig. 11.10.

Fig. 11.9 Universal quantification example

```
(exists
  (?event ?hay ?horse)
  (and
    (instance ?horse Horse)
    (instance ?event Eating)
    (instance ?hay Hay)
    (patient ?event ?hay)
    (agent ?event ?horse)))
```

Fig. 11.10 Universal quantification example

```
(forall (?boy)
  (exists (?fudge)
    (and
      (instance ?boy Man)
      (instance ?fudge Food)
      (enjoys ?boy ?fudge))))
```

11.4.8 Possessives

Possessive statements are of the form “X’s Y”, or, less fluently, “The Y of X.”. The type of the arguments changes the form of the relationship between the entities. “Tom’s father is rich.”, “Bob’s nose is big.”, “Mary’s car is fast.” See their respective CELT translations in Fig. 11.11. The second example shows a case of where no

```
(exists
  (?father)
  (and
    (attribute Tom-1 Male)
    (instance Tom-1 Human)
    (attribute ?father Rich)
    (father ?father Tom-1)))

(exists
  (?nose)
  (and
    (attribute Robert-1 Male)
    (instance Robert-1 Human)
    (attribute ?nose
      SubjectiveAssessmentAttribute)
    (instance ?nose Nose)
    (part ?nose Robert-1)))

(exists
  (?car)
  (and
    (attribute Mary-1 Female)
    (instance Mary-1 Human)
    (attribute ?car Fast)
    (instance ?car Automobile)
    (possesses Mary-1 ?car)))
```

Fig. 11.11 Possessive examples

equivalent term has been created in SUMO for the adjective “big”. This also raises a general issue that some words in natural language are sufficiently vague that it is not possible to state a formal logical meaning for them, at least not without a more sophisticated approach than relating them to a single formal term.

11.4.9 Anaphor

CELT handles simple pronoun references, including those across multiple sentences. The user can chose to have CELT process any number of sentences at one time into a single (possibly large) logical expression. It also handles some references on the basis of class descriptions. The example “The man drives to the store. He buys cookies.” yields the translation shown in Fig. 11.12. CELT can also handle possessive pronoun references, such as “Bob has a house. Mary likes his cat.” (Fig. 11.13).

Fig. 11.12 Anaphor example

```
(exists
 (?cookies ?event1 ?event2 ?man ?store)
 (and
  (forall
   (?cookie)
   (=>
    (member ?cookie ?cookies)
    (instance ?cookie Food)))
  (instance ?cookies Group)
  (instance ?event1
   LandTransportation)
  (instance ?event2 Buying)
  (attribute ?man Male)
  (instance ?man Man)
  (instance ?store RetailStore)
  (patient ?event2 ?cookies)
  (agent ?event1 ?man)
  (destination ?event1 ?store)
  (agent ?event2 ?man)))
```

Fig. 11.13 Anaphoric reference for possessives

```
(exists
 (?cat ?house)
 (and
  (attribute Mary-1 Female)
  (attribute Robert-1 Male)
  (instance Mary-1 Human)
  (instance Robert-1 Human)
  (enjoys Mary-1 ?cat)
  (instance ?cat Feline)
  (possesses Robert-1 ?cat)
  (instance ?house House)
  (possesses Robert-1 ?house)))
```

11.4.10 Conjunction and Disjunction

Although CELT can handle simple conjunctions, more work is needed. While handling conjunction of predicates, CELT can handle shared subjects and objects. For example, the sentence “Bob and Mary cut and paste the photo and the newspaper.” is translated in Fig. 11.14. Here the assumption that every participant participates in every activity is not necessarily true but CELT takes a simplistic assumption on this issue.

Fig. 11.14 Conjunction

```
(exists
  (?event1 ?event2 ?newspaper ?photo)
  (and
    (attribute Mary-1 Female)
    (attribute Robert-1 Male)
    (instance Mary-1 Human)
    (instance Robert-1 Human)
    (agent ?event1 Mary-1)
    (agent ?event1 Robert-1)
    (instance ?event1 Cutting)
    (agent ?event2 Mary-1)
    (agent ?event2 Robert-1)
    (instance ?event2 Attaching)
    (instance ?newspaper Newspaper)
    (instance ?photo Photograph)
    (patient ?event1 ?newspaper)
    (patient ?event1 ?photo)
    (patient ?event2 ?newspaper)
    (patient ?event2 ?photo)))
```

11.4.11 Negation

Negation is a challenging issue for interpretation. While the logic expressions of two sentences “Bob wrote the book.” and “Bob wrote a book.” are similar, their negative counterparts, “Bob did not write the book.” (Fig. 11.15) and “Bob did not write a book.” (Fig. 11.16) have quite different logic interpretations. The former sentence assumes context of reference to a particular book, stating that Bob did not write it. The latter states that there exists no book that Bob wrote.

```
(exists (?book)
  (and
    (attribute Robert-1 Male)
    (instance Robert-1 Human)
    (not
      (authors Robert-1 ?book))
    (instance ?book Book)))
```

Fig. 11.15 Negation of a definite reference

Fig. 11.16 Negation of an indefinite reference

```
(not
  (exists (?book)
    (and
      (attribute Robert-1 Male)
      (instance Robert-1 Human)
      (authors Robert-1 ?book)
      (instance ?book Book))))
```

11.5 CELT Components

We use morphological processing rules, derived from the “Morphy” code of WordNet to transform other verb tenses and plural verbs into the various tenses and numbers required. Discourse Representation Theory (DRT) (Kamp and Reyle, 1993) handle context to resolve anaphoric references, implications, and conjunctions. CELT is implemented in SWI-Prolog and its grammatical rules are expressed in a Definite Clause Grammar (DCG). The DCG formalism is extended with the feature grammar extension of GULP 3.1 (Covington, 1993). Feature grammars specify features such as case, number, and gender. Thus CELT’s grammar rules form a unification grammar.

Acknowledgments The authors would like to thank the US Army and DARPA for funding the development of CELT, and Army CECOM for funding development of SUMO and the SUMO-WordNet mappings. We would also like to thank the anonymous reviewers of this chapter for their helpful comments, some of which were directly included in the text to improve explanation.

References

- Allen, J.F., L.K. Schubert, G. Ferguson, P. Heeman, C.H. Hwang, T. Kato, M. Light, N.G. Martin, B.W. Miller, M. Poesio, and D.R. Traum. 1994. The TRAINS project: A case study in defining a conversational planning agent. Technical report. UMI Order Number: TN94-3, University of Rochester.
- Covington, M. 1993. *Natural language processing for prolog programmers*. Englewood Cliffs, NJ: Prentice Hall.
- Davidson, D. 1980. *The logical form of action sentences. Essays on actions and events*, 105–22. New York, NY: Oxford University Press, ISBN: 0198246374.
- Fellbaum, C., ed. 1998. *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.
- Fuchs, N., U. Schwertel, and R. Schwitter. 1999. Attempto controlled english (ACE) language manual, Version 3.0. Technical Report 99.03, August 1999, Department of Computer Science, University of Zurich.
- Kamp, H., and U. Reyle. 1993. *From discourse to logic*. London: Kluwer Academic Publishers.
- Kucera, H., and W.N. Francis. 1967. *Computational analysis of present-day American english*. Providence, RI: Brown University Press.
- Landes S., C. Leacock, and R.I. Teng. 1998. Building semantic concordances. In *WordNet: An electronic lexical database*, ed. C. Fellbaum. Cambridge, MA: The MIT Press.
- Niles, I., and A. Pease. 2001. Towards a standard upper ontology. In *Proceedings of Formal Ontology in Information Systems (FOIS 2001)*, 17–19 Oct, 2–9. Ogunquit, Maine. See also www.ontologyportal.org

- Niles, I., and A. Pease. 2001. Origins of the IEEE standard upper ontology. Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology, 37–42. Seattle, WA.
- Niles, I., and A. Pease. 2003. Linking lexicons and ontologies: Mapping wordnet to the suggested upper merged ontology. In Proceedings of the IEEE International Conference on Information and Knowledge Engineering, 412–416. Las Vegas, NV.
- Pease, A., and W. Murray. 2003. An english to logic translator for ontology-based knowledge representation languages. In Proceedings of the 2003 IEEE International Conference on Natural Language Processing and Knowledge Engineering, 777–783. Beijing, China.
- Vossen, P., and C. Fellbaum, ed. 2002. In Proceedings of the First International WordNet Conference – GWC 2002, 19–25 Jan 2002. Mysore, India: Central Institute of Indian Languages.

Chapter 12

Cyc

Douglas Foxvog

12.1 Introduction

The Cyc project was begun in the mid-80s by Doug Lenat¹ to become an ontology of “all commonsense knowledge.” This goal was expressed in various different ways. It was often defined as all the knowledge that one would need in order to understand encyclopedia articles. For this reason the name “Cyc” was derived from “encyclopedia” and is pronounced that way. A related goal has been expressed that the Cyc knowledge base should include all the general knowledge about the world that a five-year old should know. A corollary of these goals is that the Cyc Language (CycL) must be designed to be expressive enough to express any thought that may be included in an encyclopedia and to reason about anything that can be so expressed to obtain the unstated information implicit in the text. (Lenat, 1995)

These goals have driven the design of Cyc and CycL from the beginning. Not only did the ontology have to be able to represent a vast array of types of things but it had to be able to express simple and complex relations among them. The first design of CycL was as a frame language, with named slots on class types, but as the project grew, the utility of higher-order relations became apparent and CycL was redesigned as a higher-order logical language in which statements are represented by relations between and among things represented in the language. (Pittman and Lenat, 1993)

CycL became a language using the LISP format in which statements could be made about every term in the language. LISP allowed CycL to easily represent higher-arity and variable-arity relations as well as the more common binary relations. No reserved words were syntactically restricted from being arguments to relations and being reasoned about. Every Cyc term had to be an instance of something, so the term **Thing** was created as the universal class that included

D. Foxvog (✉)
Digital Enterprise Research Institute (DERI) Ireland, Galway, Ireland
e-mail: doug.foxvog@deri.org

¹The Cyc project started as part of the Microelectronics and Computer Technology Corporation (MCC), spinning off in 1994 to become Cycorp.

everything (including itself) as an instance and which included every other class as a subclass². (Cycorp, 2002)

12.1.1 *The Form of the Language*

Terms in CycL are either atomic or of the form, (**<relation> <arg1> ... <argn>**). Atomic terms are either variables (indicated with a leading “?”) or constants (which may be terms defined in CycL, strings, or rational numbers). Relations may be either predicates, which return a truth value, or functions, whose return type should be defined. Constant atomic terms defined in CycL are normally written with a leading “#” except when long CycL expressions are being presented, in which case the “#” is usually omitted. Stylistically, the names of predicates start with lower case letters, while all other terms use upper case. Function names generally end with “Fn” (excepting units of measure) while context names generally end with “Mt” (for “microtheory”).

12.1.2 *Vocabulary*

Classes in CycL are called **#\$Collections**, while things that are not classes (or sets) are called **#\$Individuals**. **#\$Relation**, partitioned into **#\$Predicate** and **#\$Function**, is a subclass of **#\$Individual**. Predicates are normally fixed-arity (usually binary), but some are variable arity, allowing a range of number of arguments. CycL does not have a separate structure for attributes attached to classes, but uses domain-restricted **#\$BinaryPredicates** instead.

```
(#$isa #$MKGandhi #$MaleHuman)
($genls #$MaleHuman #$Human)
($genls #$MaleHuman #$MaleAnimal)
```

Being an instance of a class is represented by the (code-supported) binary predicate, **#\$isa**, while the predicate, **#\$genls** (short for “generalizations”), is used to express the subclass relation³. Membership in a set is represented by the binary predicate, **#\$elementOf**. More complex logical relations, such as **#\$implies**, **#\$forall**, and **#\$thereExists**, are also instances of **#\$BinaryPredicate**. Logical connectors such as **#\$and** and **#\$or** are **#\$VariableArityPredicates**, allowing an indefinite number of arguments. The Cyc inference engine has special modules for each of these (and many more) predicates.

Collections are similar to, but distinct from, sets (**#\$Set-Mathematical**). A set is defined by its members – membership in a set is fixed for all time – while membership in a class, e.g. **#\$HumanChild**, can be different at different times. It is possible for two different collections, e.g. **#\$HomoSapiens**

²See Section 12.2.1 below.

³A class is considered a subclass of itself.

and `#$GenusHomo`, to have the exactly the same elements in some contexts, while this cannot happen in the case of sets. Instances of a given collection all have some more-or-less significant property in common, while the elements of a given set may have nothing in common besides their membership in the set. Sets in CycL are rarely identified by a constant, normally being either extensionally specified by enumerating its elements, e.g. `(#$TheSet 3 4 5)`, or intensionally specified by a defining property, e.g. `(#$TheSetOf ?X (#$and (#$isa ?X #$Integer) (#$greaterThan ?X 42)))`. [`#$TheSet` is a variable-arity function, while `#$TheSetOf` is a binary function, both of which return an instance of `#$Set-Mathematical`.]

12.1.3 OpenCyc and ResearchCyc

Cycorp has released an open source version of Cyc, called OpenCyc⁴, with hundreds of thousands of terms, millions of assertions relating the terms to each other, the Cyc inference engine and browser, and documentation for using the system. ResearchCyc, providing programming tools, additional user interfaces, and a more powerful natural language system is available to researchers under a more restrictive licence (Fig. 12.1).

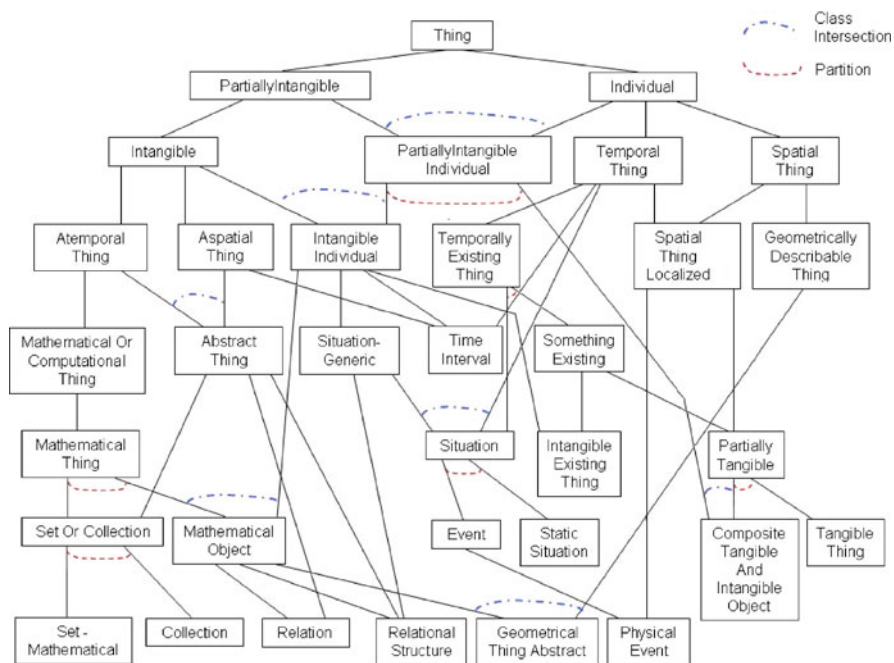


Fig. 12.1 Cyc upper ontology

⁴<http://opencyc.org/>

12.2 Upper Ontology

There have been many efforts to describe an upper level ontology under which all other ontologies can fall. Cyc needs to represent the concepts in each, if it is to achieve its goal of being able express and reason about “anything”. Thus Cyc has multiple ways of describing and subdividing the universal concept, **#\$Thing**, at the most general level.

Partitions of **#\$Thing** include **#\$TemporalThing** – **#\$AtemporalThing**, **#\$SpatialThing** – **#\$AspatialThing**, **#\$Intangible** – **#\$PartiallyTangible**, and **#\$SetOrCollection** – **#\$Individual**. Although some of these classes are subclasses of others, e.g. **#\$PartiallyTangible** is a subset of **#\$TemporalThing** and **#\$SpatialThing** while **#\$SetOrCollection** is a subclass of **#\$Intangible**, intersections of other of these classes bring more concepts into the upper ontology.

Each of these concepts is divided further. For example, **#\$TemporalThing** is partitioned into **#\$TimeInterval** and **#\$TemporallyExistingThing**, which itself is divided into **#\$SomethingExisting** and **#\$Situation** – the enduring and perdurant which some other upper ontologies have as their basic split. **#\$CompositeTangible-AndIntangibleObject** is defined as the set of physical objects which have some non-physical, e.g. symbolic, aspects as well. This includes anything representational, with writing, or used symbolically – even if natural, such as a boundary river.

A key subclass of **#\$IntangibleIndividual** is **#\$Relation**, which is partitioned into **#\$Predicate** and **#\$Function**. Relations have a defined number of arguments (called arity) which is normally fixed, but could be variable. The individual arguments are restricted to being instances of a specified class and to being subclasses of a specified class when appropriate. The result type of functions is similarly defined. Because Cyc uses binary predicates instead of slots or attributes attached to classes, the creation of pairs of binary predicates which differ merely in their argument order (e.g. both `hasParent` and `parentOf`) is discouraged.

```
(#$isa #$biologicalMother #$BinaryPredicate)
($arg1Isa #$biologicalMother #$Animal)
($arg2Isa #$biologicalMother #$FemaleAnimal)
```

Numerous predicates and relations, especially those which would be covered by reserved words in other logic languages, are supported by code in the Cyc inference engine. Argument restrictions for relations are defined through the use of such code-supported relations. The binary predicate **#\$arg1Isa** restricts the first argument of a relation to being an instance of the collection which is the second argument of **#\$arg1Isa**. Assertions using the relation are rejected from being put into the knowledge base (KB) if the first argument is not known to be such an instance and such a conclusion will not be derived from rules. The predicate **#\$arg2Gen1s** similarly restricts the second argument to being a subcollection of the specified collection and **#\$resultIsa / #\$resultGen1** makes any

non-atomic term generated by a function to be an instance/subclass of a specified collection [see example under Section 12.4 below]. Cyc is internally documented, with the predicate **#\$comment** used to describe each defined atomic term.

```
(#$isa #relatives #SymmetricBinaryPredicate)
($isa #biologicalMother #AsymmetricBinaryPredicate)
($genlPreds #biologicalMother #relatives)
($comment biologicalMother "A FamilyRelationSlot that
relates a given Animal to its female biological parent.
(biologicalMother KID FEMALE) means that FEMALE
is the female biological parent of KID. Cf. mother.")
```

Several binary predicate types are defined by code-supported logical features. A binary predicate, e.g. **#\$relatives**, is symmetric if (**pred ?A ?B**) implies (**pred ?B ?A**) and asymmetric if (**pred ?A ?B**) implies (**not (pred ?B ?A)**), e.g. **#\$olderThan**. A binary predicate, e.g. **#\$greaterThanOrEqualTo**, is reflexive if (**pred ?A ?A**) holds for all permissible arguments and irreflexive if (**not (pred ?A ?A)**) does, e.g. **#\$perpendicularObjects**. Similar predicate types are transitive and anti-transitive. **#\$genlPreds** indicates that one predicate is “more general” than another, i.e., its second argument holds as a predicate for a set of arguments in all cases in which its first argument does.

```
(#$functionalInArgs #biologicalMother 2)
```

#\$functionalInArgs defines predicates as being “functional” in one or more arguments. This means that for any particular way of fixing each of the predicate’s other arguments, there will be at most one thing that, if taken as the *n*th argument, would result in a true sentence. For example, **#biologicalMother** is functional in its second argument, as nothing has more than one biological mother. Code support prevents statements with a different value for the functional argument from being asserted or deduced.

Cyc formerly defined **#\$Attribute** [think “attribute value”] as a key subclass of **#\$IntangibleIndividual**, for the ranges of those binary predicates which do not seem to be either classes or normal individuals, e.g. **#\$Color** was a subclass of **#\$Attribute** and **#\$MaroonColor** was an instance. Cycorp later started restructuring instances of **#\$Attribute** as classes of things possessing those attribute values. Therefore, **#\$MaroonColor** now represents the class of all spatial things possessing that color. However, attribute values that can be specified as scalar ranges, such as hardness have remained as individuals, resulting, e.g. in **#\$HardAsARock** being an individual, and not meaning the class of all objects with the specified hardness.

Cyc uses **#\$isa** both for being necessarily an instance of some class (such as a person) and being an instance of a class (such as boy) in the current context.

12.2.1 Higher Order Classes

Because in Cyc, everything is an instance of several classes, the concept of “instance” has not been made disjoint from classes. “Instance” is a distinction of the language used for modelling, not one in the world being modelled. The distinction that Cyc makes is between classes, called **#\$Collections**, and things that cannot be classes or sets, called **#\$Individuals**. Sets (**#\$Set-Mathematical**), as described above, are little used.

Ontology languages normally distinguish types of objects (also called class, concept, collection, etc.) from things that are not types (individuals, instances, etc.). An issue that has often been recognized is that types of classes exist, which means that instances of these meta-classes are themselves classes. Some systems address this issue by disallowing meta-classes, some allow meta-classes without restriction, and others do not distinguish between classes and individuals. In order to provide for rigor in this field, an ontology of levels of meta-classes (**#\$CollectionType**) was created for the Cyc Project (Foxvog, 2005). [See Fig. 12.2.]

Classes are distinguished by their “order” – the number of iterations of instantiations that are necessary in order to obtain individuals.

First-Order Collection: The meta-class of all subclasses of **#\$Individual**. Each instance of **#\$FirstOrderCollection** is a class, each of whose instances is an individual. This is the most common type of class, having instances such as **#\$Person**, **#\$Computer**, and **#\$Ontology**. The class, **#\$Individual**, is an instance of **#\$FirstOrderCollection** since, by definition, all of its instances are individuals.

Second-Order Collection: The meta-class of all subclasses of **#\$FirstOrderCollection**. Each instance of **#\$SecondOrderCollection** is a class, each of whose instances is a first-order collection. Typical Second-Order classes include **#\$CarBrand**, **#\$AnimalSpecies**, **#\$PersonTypeByOccupation**, and

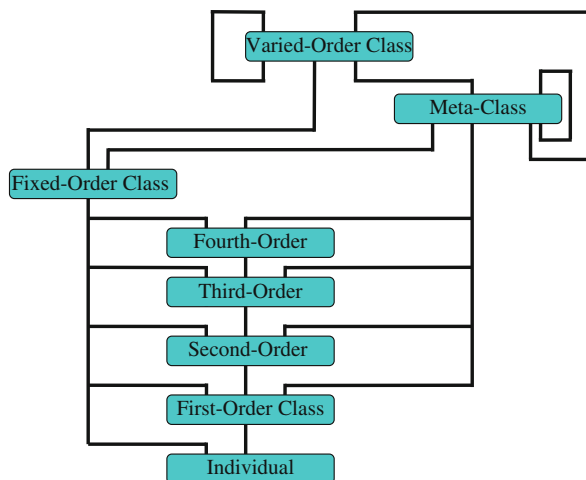


Fig. 12.2 Instance-of relations in meta-class ontology

#\$USArmy-Rank. **#\$FirstOrderCollection** is an instance of **#\$SecondOrderCollection** since, by definition, all of its instances are first order.

Third-Order Collection: The meta-class of all subclasses of **#\$SecondOrderCollection**. Each instance of **#\$ThirdOrderCollection** is a class, each of whose instances is a second-order collection. Typical Third-Order classes include **#\$BiologicalTaxonType** and **#\$MilitaryRankType**. **#\$SecondOrderCollection** is an instance of **#\$ThirdOrderCollection** since, by definition, all of its instances are second order.

Fourth-Order Collection is the meta-class of all subclasses of **#\$ThirdOrderCollection**. Higher order meta-classes could be similarly defined; however, the utility of implementing such classes is questionable. The only fourth-order class in OpenCyc (OpenCyc) is **#\$ThirdOrderCollection**. Similarly, **#\$FourthOrderCollection** would likely become the only instance of Fifth-Order Collection, and so on, *ad infinitum*.

#\$Individual, **#\$FirstOrderCollection**, **#\$SecondOrderCollection**, and **#\$ThirdOrderCollection**, are mutually disjoint classes, which each have the property that every one of their instances has the same order, i.e. it takes a fixed number of iterations of instantiation to reach an Individual. Thus, every instance of Individual is an Individual – this could be considered zero-order. No instance of First-Order Collection is an Individual, but every instance of every instance of it is. No instance or instance of an instance of Second-Order Collection is an Individual, but every instance of an instance of an instance of it is. Likewise, for Third- and Fourth-Order Collection.

Fixed-Order Collection is defined as the meta-class of all classes with this property. **#\$FirstOrderCollection**, **#\$SecondOrderCollection**, **#\$ThirdOrderCollection**, and **#\$FourthOrderCollection** are not only instances of **#\$FixedOrderCollection**; they are subclasses of it as well. **#\$Individual** is an instance, but not a subclass of **#\$FixedOrderCollection**.

Not every class is a Fixed-Order Collection. **#\$FixedOrderCollection**, itself, has classes of different orders as instances. Thus, it is considered a varied order collection. *Varied-Order Collection* is defined as the class of all classes which (1) have instances of more than one order, (2) are instances of themselves, or (3) have any instances of variable-order. Every class is therefore either fixed-order or variable order, but not both.

#\$VariedOrderCollection is an instance of both itself and **#\$CollectionType**. **#\$Collection**, **#\$CollectionType**, and the universal class (**#\$Thing**) are instances of **#\$VariedOrderCollection**. **#\$VariedOrderCollection** is not a subclass of **#\$CollectionType** since some of its instances, e.g. **#\$Thing**, have instances that are individuals as well as instances that are classes.

Special modules in the inference engine reason about membership in collections and are designed to block paradoxical reasoning that would naturally arise from such a structure.

12.3 Contexts

Two major problems with the concept of a global ontology, or an ontology upon which everyone can build, are that different ontologies will disagree on facts, and that most projects do not need the baggage of hundreds of thousands of terms which are not applicable to their area of interest.

In order to address such issues, as well as to divide the growing ontology into more manageable chunks, a system of contexts was designed. Just as there is a class of which every other class is a subclass, a base context (**#\$BaseKB**)⁵ is defined of which every context is a subcontext. The Cyc term for the contexts which it reifies is **#\$Microtheory** (abbreviated “Mt”) – which is a subclass of **#\$Individual**.

Below the base context, lies a directed acyclic graph (DAG) of microtheories. CycL uses the transitive binary predicate **#\$genlMt** to relate a microtheory to more general microtheory, whose statements it (in effect) inherits.⁶ An individual microtheory may have numerous direct **#\$genlMts** as well as being the direct **#\$genlMt** of numerous other microtheories.

Although assertions are not copied to a microtheory from its **#\$genlMts**, they are just as accessible as assertions made in the microtheory. This means that adding assertions to a microtheory will affect all of its “specMts”.⁷

12.3.1 Dimensions of Context Space

Contexts are not defined solely on the basis of topic. Although the basic terms being used for two different purposes can be the same, the rules about how the things represented by the terms interact, or the data expressed using that vocabulary are often in conflict. A set of dimensions of conflict space were distinguished, and the microtheory structure has been developed with these in mind (Lenat, 1998).

12.3.2 Vocabulary/Theory/Data Contexts

Microtheories have been distinguished into **Vocabulary Microtheories** which define the general terms used for some topic, **Theory Microtheories** which express rules about how the things represented by the vocabulary behave, and **Data Microtheories** which provide information, using vocabulary and theories defined in other microtheories, about events and individuals in some specific context. Data

⁵For the sake of efficiency and manageability, the **#\$BaseKB** has been separated into a set of mutually accessible contexts for the use of internal reasoning modules.

⁶A microtheory is considered a **#\$genlMt** of itself. This is an exception to the **genlMt** graph being a DAG.

⁷The term “spec” – short for “specialization” – is Cyc terminology for the inverse of “genl” – short for “generalization”.

microtheories can be considered to be knowledge bases, while vocabulary and theory microtheories contain ontologies.

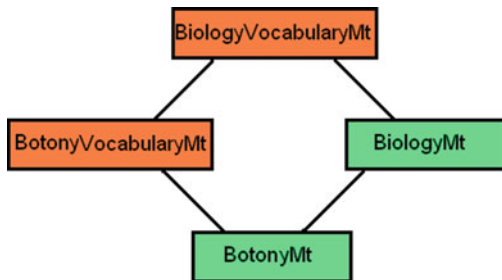
More than one theory microtheory may share the same vocabulary microtheory, but provide different rules. For example, rules for taxes or business incorporation vary by jurisdiction, the properties of telephones are different now than they were a decade ago, and physics reasoning is different for children, average adults, engineers, and relativistic scientists.

Data microtheories may differ by vendor (the price of a book), by date (whether Montenegro is an independent country), by location (the sun is directly north at 12:01 p.m. GMT, 6 November 2008), or by conceptual work (Oceania is at war with Eurasia) although they share the same vocabulary (except for mt-specific individuals) and rules.

Vocabulary microtheories are designed to have only other vocabulary microtheories as **#\$genIMts**; theory microtheories can have both vocabulary and theory microtheories as **#\$genIMts**; while data microtheories can have any of these three types as **#\$genIMts**. If two theory or data microtheories each have an associated vocabulary microtheory and a **#\$genIMt** assertion is made between them, a **#\$genIMt** assertion is automatically concluded between their associated vocabulary microtheories. [See Fig. 12.3] However, the restrictions on microtheory content by microtheory type are not enforced by code.

Cyc-based projects are expected to create context-specific microtheories for their specific use cases with appropriate general purpose mts attached as genIMts. Project-specific data is added to these mts and queries are carried out in these project-specific contexts – or in more specialized microtheories. Having such mts inherit assertions from selected general-use mts (and transitively all their genIMts) instead of placing new data in existing generic contexts prevents assertions made for specific cases from interfering in unrelated contexts.

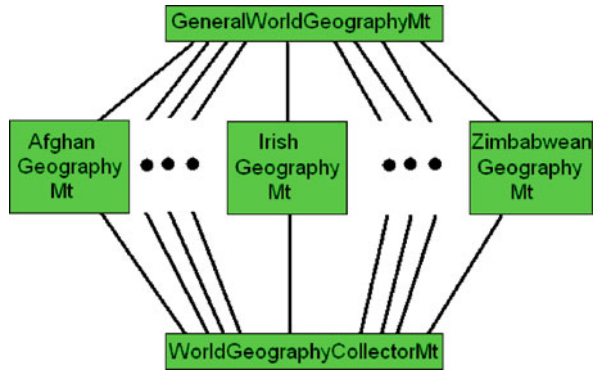
Fig. 12.3 Vocabulary and theory microtheories



12.3.3 Spindles

Microtheory spindles have been created to deal with the situation in which some topic can be broken down into a number of separate but similar topics. For example, consider the case of geographical data microtheories. A general world geography

Fig. 12.4 Microtheory spindle



knowledge base can contain basic information about the countries of the world, specific data microtheories for each country's geographical details can have that context as a `#$gen1Mt`, and collector microtheories can have all the country-specific microtheories for a given region or alliance (or the whole world) as `#$gen1Mts`. [See Fig. 12.4]

12.3.4 Problem Solving Contexts

Temporary contexts, called **Problem Solving Contexts** (PSC), can be set up for asking and answering questions. Such contexts generally define a set of contexts to use for answering the question, which are all defined as `#$gen1Mts` of the problem solving context, so that statements that are true in any of the selected contexts are (effectively) inherited (and thus true) in the PSC and reasoning has access to all such assertions at the same time.

12.3.5 Hypothetical Contexts

A **Hypothetical Context** is a problem solving context in which some statements are asserted to be true so that conclusions can be drawn from them. Such contexts may be created by special tools and automatically deleted after the tasks are completed. Both Problem Solving Contexts and the more specific HypotheticalContexts have specialized code support.

12.3.6 Fictional Contexts

Some contexts are made up, or fictional. The statements in these contexts might be true in a work of fiction or in some belief system, but are not considered true outside that work or belief system. Describing these contexts as fictional aids in reasoning about the contexts *from external contexts* and can prevent accidentally creating a more specialized context in which both the external context and the one that considers it fictional are inherited.

One cannot label microtheories as fictional in the **#\$BaseKB** since that assertion would be inherited by the microtheory itself. Two belief system contexts could each label the other fictional: e.g. if each believed that its god was the one true god, and that the other belief system worshiped a false god.

12.3.7 *UniversalVocabularyMt*

The definition of terms in microtheories below the **#\$BaseKB** led to problems in reasoning since in contexts in which terms were not relevant, they were not defined. A **#\$UniversalVocabularyMt** was created to hold partial definitions of all terms to obviate resultant problems in the inference engine.

The use of this microtheory is intended, for example, to prevent one ontologist from using a term as an individual in one context and a second ontologist unknowingly using it as a class in another. Violations of other disjointnesses such as between first- and second-order classes or binary and ternary predicates are prevented.

A decision was made to place this microtheory, not in an area which the inference engine can privately check, but in a **#gen1Mt** loop with **#\$BaseKB**, so that everything that appears in it is true in every context, no matter how unrelated.

Some classes have been defined as being so basic that any assertion that something is an instance or subclass of them are automatically lifted from whatever microtheory the assertion is made to the **UniversalVocabularyMt**. These are called **#\$Atemporal-NecessarilyEssentialCollectionTypes** or ANECTs. However, if the term is defined to be an instance or subclass of a far narrower class in a more specific context, the system does not determine the associated ANECTs that should be stated in **#\$UniversalVocabularyMt**, which can lead to reasoning problems if no **#\$isas** (or **#\$genls** for classes) for the term are found there. This has led to ontologists placing assertions in **#\$UniversalVocabularyMt** which are not true in all possible worlds.

12.4 Functions

The result of applying a function to a set of arguments is an instance of a class specified by a **#\$resultIsa** predicate, and if the result is a class, it is a subclass of a class specified by a **#\$resultGenls** predicate. The relationship between the result of the function and its arguments is specified using the predicate **#\$function-CorrespondingPredicate-Canonical**.

Some Cyc functions have code support for evaluating the value of the function term applied to its arguments, in which case they are called **#\$EvaluatableFunctions**. The result of evaluating the application of an **#\$EvaluatableFunction** to a set of arguments is a term calculated by code, e.g. (**#\$PlusFn 1 2**) results in the term '3'.

The result of applying a non-evaluatable function to a set of arguments is a non-atomic CycL term. Many non-evaluatable Cyc functions are “reifiable”, meaning that their use creates Non-Atomic Reified Terms (NARTs) which are indexed in the Cyc

knowledge base similar to constants. **#\$MotherFn** is a reifiable function defined as follows:

```
(isa MotherFn UnaryFunction)
(isa MotherFn ReifiableFunction)
(arg1Isa MotherFn Animal)
(resultIsa MotherFn FemaleAnimal)
(functionCorrespondingPredicate-Canonical MotherFn
  biologicalMother 2)
(comment MotherFn "This function maps animals to their
  biological mothers. (MotherFn ANIMAL) is the biological
  mother of ANIMAL.")
```

Thus, (**#\$MotherFn** **#\$MKGandhi**) would be a term in the Cyc KB which represents a female such that (as implied by the fifth line above):

```
(biologicalMother MKGandhi (MotherFn MKGandhi)).
```

Instances of **#\$UnitOfMeasure** and **#\$DateDenotingFunction** are types of non-evaluable functions for which reification is not deemed useful. Their instances still create non-atomic CycL terms, such as (**#\$Meter 27**) and (**#\$YearFn 1984**), but these are treated differently, e.g. not indexed, in the Cyc knowledge base.

12.4.1 Prototypes

Cyc uses a function called **#\$The** to express prototypes, greatly simplifying expressions. The technique was developed for describing relations among body parts in a prototype human (or other animal) enabling corresponding relations to be concluded about specific humans. In the anatomical field, **#\$The** may only be applied to part types of which there are only one, i.e. instances of **#\$UniqueAnatomicalPartType**. (Lehmann and Foxvog, 1998)

For example, one can assert,

```
(pipeEndsAtCavity (The Aorta)
  (The (LeftObjectOfPairFn VentricleOfHeart)))
```

in **#\$HumanPhysiologyMt**, from which, given **#\$Aorta** being an instance of **#\$UniqueAnatomicalPartType** and **#\$VentricleOfHeart** being an instance of **#\$SymmetricAnatomicalPartType** (resulting in (**#\$LeftObjectOfPairFn** **#\$VentricleOfHeart**) being an instance of **#\$UniqueAnatomicalPartType**), Cyc can conclude that any person's aorta is connected to that same person's left ventricle:

```
(pipeEndsAtCavity (UniqueBodyPartFn MKGandhi Aorta)
  (UniqueBodyPartFn MKGandhi
    (LeftObjectOfPairFn VentricleOfHeart)))
```

12.4.2 Skolemization

Skolemization is a technique of removing existential quantifiers from a formula, replacing the existential term with a function of all the free terms. Cyc has a special class of functions to perform this conversion called **#\$SkolemFunction**, which is a class of system-generated functions used to implement existential quantification. Whenever a sentence that is headed by **#\$thereExists** is asserted, the inference engine creates a new instance of **#\$SkolemFunction** and modifies the assertion to use that function. For example, the rule

```
(implies
  (isa ?ANIMAL Vertebrate)
  (thereExists ?MOM
    (mother ?ANIMAL ?MOM)))
```

is converted to

```
(implies
  (and
    (isa ?ANIMAL Vertebrate)
    (termOfUnit ?MOM (SKF-123456 ?ANIMAL)))
  (mother ?ANIMAL ?MOM)))
```

along with the definition of the Skolem Function.

12.5 Reasoning

The Cyc inference engine has a large number of specialized reasoning modules in addition to general modules for handling IF-THEN implications. Some of the publicly acknowledged general features of the inference engine are described in this section.

12.5.1 Forward and Backward Chaining

Rules in Cyc are of the form (**#\$implies SENTENCE1 SENTENCE2**) and are declared as either forward or backward chaining. A forward chaining rule is continuously active such that whenever the inference engine is presented with (or concludes) a sentence matching **SENTENCE1**, with all arguments bound, the conclusion is made. For example, a forward-chaining rule on **#\$biologicalMother** could be used to conclude that a mother is the same species as the child:

```
(implies
  (and
    (biologicalMother ?KID ?MOM)
    (isa ?KID ?SPECIES)
```

```
(isa ?SPECIES BiologicalSpecies)
(isa ?MOM ?SPECIES))
```

A backward chaining rule would only activate when a request was made to the inference engine that matched **SENTENCE2**. In this case, the inference engine would try to prove **SENTENCE1** with all variables that were bound in **SENTENCE2** bound the same way in **SENTENCE1**.

12.5.2 Don't Care Variables

Normally, any variables that appear in the antecedent of a rule would be expected to appear in the consequent or multiple times in the antecedent. However, sometimes only the existence of something that fills a position in some relation is needed to make the conclusion. Instead of requiring the rule to be written with **#\$thereExists** in the antecedent, Cyc permits the rule to be written with what is called a “don't care variable”, which occurs only once in a rule – in the header. Such a variable is denoted by an initial “??”. For example:

```
(implies
  (biologicalSons ?DAD ??SON)
  (isa ?DAD Parent))
```

12.5.3 Rule Macro Predicates

As hundreds of thousands of rules were being asserted into Cyc, a large number of standard patterns appeared. As these were detected, special predicates were designed to stand as macros for the rule patterns, code modules were written for the inference engine to speed up reasoning, and the rules were replaced by the assertions using these “rule macro predicates”. For example,

```
(interArgCondIsa1-2 anatomicalParts Vertebrate
  Head-AnimalBodyPart Head-Vertebrate)
```

uses the rule macro predicate, **#\$interArgCondIsa1-2** to mean:

```
(implies
  (and
    (isa ?VERT Vertebrate)
    (anatomicalParts ?VERT ?HEAD)
    (isa ?HEAD Head-AnimalBodyPart)
    (isa ?HEAD Head-Vertebrate))).
```

The equivalent rule is not asserted in the knowledge base, but the assertion in the macro form is used far more efficiently for reasoning.

It turns out that the predicate for expressing the subclass relationship is a rule macro predicate, since **(genls ?A ?B)** means **(implies (isa ?I ?A) (isa ?I ?B))**.

12.5.4 Monotonic vs. Default Reasoning

Each statement in a Cyc knowledge base is asserted to be either monotonically true or default true. Conclusions derived solely from monotonically true statements are also monotonically true. Conclusions based on at least one default true statement are only default true. If defeasible reasoning is turned on, incompatible conclusions may be returned in answer to a query, using statements that are default true, with the querier being able to view the pro and con arguments. A monotonic conclusion would negate any inconsistent default conclusions.

12.5.5 Exceptions to Rules

Cyc allows exceptions to be stated for default rules. Multiple exceptions may be stated for the same rule. The forms are asserted as:

```
(exceptWhen STATEMENT RULE)
```

and

```
(exceptFor TERM SINGLE_VARIABLE_RULE).
```

The standard usage of **#\$exceptWhen** is for **STATEMENT** to have one or more variables which are present in **RULE**. For any bindings for these variables in which **STATEMENT** is true the rule will not be considered. This form can also be used to prevent a rule that is inherited from a more general microtheory from being considered in the microtheory in which the exception is stated. An example of the standard usage (in **#\$Terrestrial-FrameOfReferenceMt**) is:

```
(exceptWhen
  (spatiallySubsumes ?GEOTHING NorthPoleOfEarth)
  (implies
    (isa ?GEOTHING GeographicalThing)
    (northOf NorthPoleOfEarth ?GEOTHING)))
```

The second exception predicate is used to express that a rule with a single variable in it is not considered for the variable being bound to the specified term. For example:

```
(exceptFor
  Taiwan-RepublicOfChina
  (implies
    (isa ?X ChineseProvince)
    (geopoliticalSubdivision China-PeoplesRepublic ?X)))
```

Statements asserted using these predicates are converted to more complex forms using another predicate, **#\$abnormal**, upon being asserted.

12.6 Events

Events and static situations are represented in Cyc as temporal objects in their own rights. This allows an unlimited number of things to be asserted (or concluded) about the event, which would not be possible if the action were represented by a predicate which related the various roles. Since this issue was brought up by philosopher Donald Davison in a 1967 paper, modelling of events as individuals has been called Davidsonian representation.

12.7 Conceptual Works

Another ontological issue that Cyc had to deal with was the representation of conceptual works. For example, what is *Romeo and Juliet* by William Shakespeare? Is it a class of things, whose instances include performances on stages, movies, pamphlets, books and sections of books, and readings given before audiences? Or is it a text string? Or a context in which certain things are true? Does it have start and end times?

Cycorp handled all of these concepts by developing a model of conceptual works. A **Conceptual Work** is defined as an intangible artifact with a start time (when it was created), possible physical instantiations or “embodiments” (not “instances”) in objects and/or events, and associated data structure(s) and context(s). If a conceptual work expresses a meaning it is considered to be a **Propositional Conceptual Work** (PCW).

Conceptual works can be instantiated multiple times in **Information Bearing Things** (IBTs) which can be physical objects or events. IBTs include physical books, performances of a play, movie performances, the Mona Lisa hanging in the Louvre, an image of the Mona Lisa on a computer screen, and a computer file on a disc from which the image on the screen was generated.

Most conceptual works have associated **Abstract Information Structures** (AISes), which are intangible conceptual structures of symbols (text strings, word strings, symphonic scores, maps, pixels, . . .) with associated communications conventions. The creator of the conceptual work defines the structure of the work, although technically s/he does not create the structure. In some philosophical sense, such symbol patterns have existed for all time – or at least since their included symbols have been defined. The information in an AIS may be specified to different degrees (words, characters, characters in coding systems (e.g. ASCII vs. UNICODE), font, color, size, . . .). AISes are individuals which can be instantiated in physical objects or physical events (but such IBTs are not their “instances”).

An IBT can instantiate multiple AISes. For example, a spoken Serbo-Croatian word instantiates both Latin and Cyrillic text strings, and the same bits in a computer instantiate the binary string “1011”, octal “13”, decimal “11” and hex “B”.

Combinations of a single symbol string with different communication conventions are different abstract information structures. For example, the string “11” denotes a different AIS if the communication convention is binary, octal, decimal, or hex, and the string “sin” denotes a different AIS in Spanish, English, and Swedish.

A sibling concept to conceptual work is **Game**, which has many similar properties: instantiations, structures for the instantiations, creators, versions, etc. Game and Conceptual Work are subsumed by the concept **Devised Practice Or Work**, which has other subclasses such as **Award Practice**, as well.

12.8 Open/Closed World Assumption

The Cyc inference engine normally operates on an open-world assumption, concluding that a statement is false only if it can be proved false and making no conclusion as to the truth of a statement that it can prove neither true nor false. Cyc also provides ways to allow reasoning using a closed world assumption – that any (non-negated) sentence that cannot be proved true is considered to be false. The most basic way to do this is through the unary predicate **#\$knownSentence** which means that the sentence is trivially provable by the Cyc inference engine from sentences currently accessible from the context in which it is asked. If **#\$knownSentence** is not true for a sentence in a given context, then it is false. **#\$knownSentence** can be used in a query or the antecedent to a rule. **#\$knownSentence** can not be asserted true or false for any sentence (because the unary predicate **#\$notAssertible** applies to it), it can only be concluded, and such a conclusion is not cached.

Several unary predicates permit a closed word assumption in a given context regarding specific predicates or membership in a certain class. The predicate **#\$complete-ExtentDecidable** indicates that the inference engine can determine the truth of any fully-bound sentence headed by that predicate. The predicate **#\$completely-DecidableCollection** indicates that Cyc can determine if any given term is a member of a specified class. In both cases, Cyc will conclude that matching sentences are false if it can not prove that they are true. Narrower predicates, **#\$completely-AssertedCollection** and **#\$completeExtentAsserted**, are used to indicate that all matching true assertions are already present in the knowledge base, so that no backchaining on rules is necessary to determine the truth of a matching assertion, merely lookup. Such assertions are normally context dependent:

```
(completelyAssertedCollection BritishColony)
(completeExtentAsserted colonyOf)
(completeExtentDecidable colonialPowersFlag)
```

but sometimes they are true throughout the KB:

```
(completelyDecidableCollection Integer)
```

These predicates are code supported and context dependant.

12.9 Geopolitical Entities

One distinction that is sometimes important is that of geopolitical entities as organizations that can take actions versus their territories, which are geographical regions. The two concepts are tightly connected, but the physical region normally existed

long before the country/ city/state whose territory it is. Actions are taken by the organization, not by the land mass. At one level Cyc maintains the distinction between the two, using a function to refer to the territories of the geopolitical entities. Thus one can say:

```
(bordersOn (TerritoryFn India)
           (TerritoryFn Pakistan))
```

Cyc also has a context, **#\$DualistGeopoliticalMt**, in which the distinction is blurred and the geopolitical entity is also a geographical region. Any context which has this as a **#\$genlMt** can also refer to the physical extent of the geopolitical entity by referring to that entity. In such a “dualist” context, the above statement could be expressed:

```
(bordersOn India Pakistan).
```

12.10 Temporal Reasoning

Cyc provides several ways of performing temporal reasoning. By default, the temporal extent of a statement is that of the context in which it resides. The temporal extent of a Cyc context is defined by its time interval (specified by **#\$mtTimeIndex**) and its temporal resolution (specified by **#\$mtTimeParameter**). The time interval may be either continuous or have temporal gaps and the resolution may be continuous (**#\$TimePoint**), some coarser resolution (e.g. **#\$CalendarMinute**), or only for the specified time interval (**#\$Null-TimeInterval**). For example, if someone is talking for twenty minutes, they are not necessarily talking during each second of that time, but they are during each minute in that period, so the temporal resolution of a context holding such a statement would be greater than **#\$CalendarSecond**.

A statement with a different temporal extent than that of the context in which it is asserted can be expressed using the predicate **#\$holdsIn** or **#\$holdsSometime-During** to express the extent. For example, **(holdsIn (YearFn 1906) (residesInRegion MKGandhi Africa))** means that Mahatma Gandhi lived in Africa throughout the whole year of 1906, while **(holdsSometimeDuring (DecadeFn 191) (residesInRegion MKGandhi Africa))** means that he lived there during the first decade of the Twentieth Century (1901–1910), possibly the whole decade.

Cyc has many more temporal predicates.

12.11 Natural Language Support

Cyc has a natural language system containing a lexicon and generation rules (Burns and Davis, 1999). Words in Cyc are individuals (instances of **#\$LexicalWord**) with associated text strings, parts of speech, denotations, languages, and other properties.

Specializations of **#\$LexicalWord** have been created for several languages, but specializations for other languages are specified using the function **#\$WordInLanguageFn** to specify the appropriate specialization if it is not yet defined, e.g. **(isa Nyet-TheWord (WordInLanguageFn RussianLanguage))**.

The Cyc inference engine has modules which generate derived forms of English words given basic forms. The singular must be given for count nouns and the plural form if the word is irregular, while a mass noun only has a single form. The infinitive must be given for a verb, and any irregular forms should be specified using special predicates such as **#\$firstPersonSg-Past**. Similarly, regular forms are given for adverbs and adjectives, along with any irregular forms of comparatives and superlatives. In any cases in which a non-standard derived form is not given for an English word, the standard derived form can be generated or interpreted.

Denotations of word senses are described using the quaternary predicate **#\$denotation** followed by the word, the part of speech, the sense number for that part of speech, and the meaning of that word sense. The word sense is specified in a context for the appropriate language. The same Cyc lexical word may have word senses for different parts of speech. For example, the definition of **#\$Boot-TheWord** would include:

```
In GeneralEnglishMt
(isa Boot-TheWord EnglishWord)
(infinitive Boot-TheWord "boot")
(singular Boot-TheWord "boot")
(denotation Boot-TheWord Verb 0 Kicking)
(denotation Boot-TheWord Verb 1 BootingAComputer)
(denotation Boot-TheWord CountNoun 0 Boot-Footwear)
In BritishEnglishMt
(denotation Boot-TheWord CountNoun 1 TrunkOfCar)
```

Many short phrases act as words and are provided denotations in Cyc using predicates similar to **#\$denotation**: **#\$compoundString**, **#\$headMedialString**, and **#\$multiWordString**, (depending on whether the head word is at the beginning, middle, or end of the phrase).

Templates attached to predicates are used to generate sentences from assertions using those predicates. Multiple templates can be assigned to a single predicate.

Functional Cyc terms (such as **(#\$MotherFn #\$MKGandhi)**) can have lexical assertions automatically concluded from forward rules declared on the function. For example,

```
(implies
  (and
    (termOfUnit ?MOM (MotherFn ?KID))
    (fullName ?KID ?NAME))
  (headMedialString (TheList "the") Mother-TheWord
    (TheList "of" ?NAME) ProperNoun ?MOM))
```

Allows the conclusion that **(#\$MotherFn #\$MKGandhi)** can be written in English as “the mother of Mahatma Gandhi”.

12.12 Cyc and the Semantic Web

In order to allow the Cyc ontology to be used on the Semantic Web, Universal Resource Identifiers (URIs) have been specified for each atomic Cyc term. The form of the URI is <http://sw.cyc.com/2006/07/27/cyc/#<term>>. Functionally defined terms (NARTs) are not provided URIs.

12.13 Summary

In summary, Cyc provides a broad common-sense ontology, a higher-order language for defining and making assertions using the ontology, and an inference engine for reasoning over such a knowledge base. The inference engine provides subclass reasoning, deduction according to rules, and numerous special reasoning modules. Portions of the Cyc ontology may be extracted and reused in other projects without the inclusion of the vast majority of the Cyc ontology.

References

- Burns, K.J. and A.R. Davis. 1999. Building and maintaining a semantically adequate lexicon using Cyc. In *Breadth and Depth of Semantic Lexicons*, ed. E. Viegas, Dordrech: Kluwer.
- Cycorp. 2002. The syntax of CycL. <http://www.cyc.com/cycdoc/ref/cyclsyntax.html>.
- Davidson, D. 1967. The logical form of action sentences. In *The logic of decision and action*, ed. N. Rescher, 81–95, Pittsburgh, PA: University of Pittsburgh Press.
- Foxvog, D. 2005. Instances of instances modeled via higher-order classes. Proceedings of the Workshop on Foundational Aspects of Ontologies (FOnt 2005), 28th German Conference on Artificial Intelligence, Koblenz, Germany, Sep 2005, ISSN 1860–4471, 46–54.
- Lehmann, F., and D. Foxvog. 1998. “Putting flesh on the bones: Issues that arise in creating anatomical knowledge bases with rich relational structure”, knowledge sharing across biological and medical knowledge based systems. Papers from the 1998 Workshop: Technical Report WS-98-04, 15th National Conference on Artificial Intelligence, Madison, WI, 26 July 1998, 41–50.
- Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *CACM* 38:11, 33–38.
- Lenat, D. 1998. The dimensions of context-space, Cycorp. <http://www.cyc.com/doc/context-space.pdf>.
- OpenCyc website. <http://www.opencyc.org>, downloaded 10/2/2008.
- Pittman, K. and D.B. Lenat. 1993. Representing knowledge in CYC-9. MCC Technical Report CYC-175-93P.

Chapter 13

Ontological Foundations of DOLCE

Stefano Borgo and Claudio Masolo

13.1 Introduction

DOLCE, the *Descriptive Ontology for Linguistic and Cognitive Engineering* (Masolo et al., 2003), is a foundational ontology developed embracing a pluralist perspective: there cannot be a unique standard or universal ontology for knowledge representation. Once it is accepted that the so-called “monolithic approach” is unrealistic, it becomes clear that the different foundational ontologies must be mutually transparent by making explicit their ontological stands and formal constraints: this is necessary to make ontology interaction possible and reliable. Roughly, it is expected that an ontology is, on the one hand, philosophically well founded (by adopting a clear ontological perspective) and, on the other hand, that it provides the information for its correct application and use (for instance, by describing explicitly the basic assumptions and the formal constraints on which it relies). A consequence of this view is that, whenever a foundational ontology does not make an *explicit* commitment with respect to an ontological topic, it is assumed that the ontology is consistent with alternative ontological positions in that topic (in some cases, it may even allow coexistence of these via techniques like parametrization). This general view is quite demanding and requires a careful analysis of the ontology content and structure; DOLCE has been one of the first ontologies explicitly built to follow (and exemplify) this approach.

Regarding the content of the ontology, the aim of DOLCE is to capture the intuitive and cognitive bias underlying common-sense while recognizing standard considerations and examples of linguistic nature. DOLCE does not commit to a strong referentialist metaphysics (it does not make claims on the intrinsic nature of the world) and does not take a scientific perspective (it is not an ontology of, say, physics or of social sciences). Rather, it looks at reality from the mesoscopic and

S. Borgo (✉)

Laboratory for Applied Ontology, National Research Council, Institute of Cognitive Sciences and Technologies – ISTC–CNR; KRDB, Free University of Bolzano, Via alla Cascata, 56C, 38123 Povo-Trento (TN), Italy
e-mail: borgo@loa-cnr.it

conceptual level aiming at a formal description of a particular, yet fairly natural, conceptualization of the world.

Technically, DOLCE is the result of a careful selection of constraints (principles, primitives, axioms, definitions, and theorems) expressed in a rich logical language, namely first-order logic, in order to guarantee expressiveness, precision, interoperability, and simplicity of use. These claims are sustained by the accompanying documentation that provides information on the ontological choices and the motivations for both the structure and the formalization of DOLCE.

Due to the introductory nature of this paper (and the limited space available), this work describes and formalizes only the most general categories of the DOLCE ontology. We advise the reader that what is presented here departs in some aspects from the original DOLCE (Masolo et al., 2003): indeed in these pages we actually discuss a new and improved version of the DOLCE kernel that we call DOLCE-CORE (Borgo and Masolo, 2009).

13.2 A Bit of History

DOLCE is part of the *WonderWeb* project¹. The aim of this project is to develop the infrastructure required for the large-scale deployment of ontologies as the foundation for the Semantic Web. On the one hand, this goal involves the establishment of a Web standard ontology language and related ontological engineering technology, on the other the development of a library of foundational ontologies reflecting different ontological choices. DOLCE, which came out in 2002, is one of the ontologies included in the WonderWeb library and, at the time of writing, it is also the most developed. It has been constructed as an ontology of particulars with a clear cognitive bias: the categories have been explicitly characterized as “cognitive artifacts ultimately depending on human perception, cultural imprints and social conventions” (Masolo et al., 2003, p. 13). So far the ontology has not undergone changes² while extensions have been proposed to cover more closely some application domains. Over the years, DOLCE has been tested in several projects ranging over a variety of areas as manufacturing, business transaction, insurance services, biomedicine, multimedia, social interaction, linguistics, and the Semantic Web at large.³

The real use of the ontology in application projects is increased by the alignment with WordNet (Gangemi et al., 2003) which provided a basis to study the relationship between ontologies and linguistic resources (Prevot et al., 2005). The ontology is publicly distributed in several formats⁴ like first-order logic FOL (including

¹<http://wonderweb.semanticweb.org>

²The version we present here can be considered as the first proposal to update the ontology and it comes after almost 6 years of experience in applying it.

³See <http://www.loa-cnr.it/DOLCE.html> for a list of institution and projects that are using or have expressed interest in the DOLCE ontology.

⁴The different versions of DOLCE can be downloaded from <http://www.loa-cnr.it/DOLCE.html>. The main version is in first-order logic. Versions in other languages have been produced approx-

KIF), OWL, DAML+OIL, LOOM, and RDFS. It is also available in the Common Algebraic Specification Language,⁵ CASL, via the Hets extension⁶ which makes available theorem provers and graphical devices.

13.3 Ontological vs. Conceptual Level

To understand the DOLCE view, we should begin with the distinction between ontological entities and conceptual entities. Entities in the first group exist in the “real” world independently of our mind. The latter group comprises entities that are the result of conceptual processes (typically over ontological entities). Generally speaking, this distinction is important to understand the different modeling choices on which theories rely: for example, *event theories of time* build intervals and instants of time from temporal relationships between events. That is, in these theories intervals and instants (times in general) are taken to be conceptual entities while events are ontological entities.

Technically, a disagreement on the ontological-conceptual classification of some type of entity indicates an inhomogeneity (or, better, heterogeneity) among theories. The usual (strong) reading of Quine’s principle “to be is to be the value of a variable” highlights a sharp separation between ontological and conceptual entities with the consequence that, for instance, *times* should be banned from the domain of quantification whenever they are conceptually constructed from events. Clearly, it is not possible to do justice of common-sense language about time with such a position where times are expunged from the formal theory.

In order to make possible the comparison (and perhaps the interaction) of heterogeneous ontological options at the syntactic level within a unified formal framework (namely, within FOL), we adopt a soft reading of the Quinean principle and assume that entities in the domain of quantification can be of ontological or of conceptual nature. That is, a claim of type “ $\exists x\varphi$ ” in the formal ontology is not necessarily addressing the ontological/conceptual status of the entities satisfying φ . In this way one can include, say, both events and times in the domain of quantification without committing to them ontologically. It is then possible to formally relate these kinds of entities within the theory itself. Furthermore, note that this choice allows us to avoid the problems of reductionism which are particularly critical in foundational ontologies. Indeed, as Heil notices:

How, for instance, could we hope to re-express truths about the global political consequences of a decline in the GNP of Eastern Europe in terms of interactions among

inating the content of the FOL version by taking into account the different expressive powers of the other languages.

⁵<http://www.brics.dk/Projects/CoFI/CASL.html>

⁶From http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/index_e.htm: Hets is a parsing, static analysis and proof management tool combining various tools for different specification languages.

fundamental particles? Even if such a reduction were possible, however, it would be self-defeating. Important higher-level patterns and relations are invisible to physics. (Heil, 2005, p.31)

13.4 Properties

Once one has established how to consider entities in the domain of the ontology, she has to decide how to *describe* (and thus differentiate) entities, i.e., how to deal with properties. The nature of properties,⁷ the explanation of what it means that an entity has a property, and, more specifically, of how different entities can have the *same* property, have been widely discussed and investigated, see Armstrong (1989), Loux (1976), and Mellor and Oliver (1997) for exhaustive surveys. Moreover, *persisting* entities (i.e. entities that exist at different times) can *change* through time by having different properties at different times: *a* may have property *F* (say, “being red”) at time t_1 and an incompatible property *G* (say, “being green”) at t_2 . The nature of time and the way entities persist and change through time are topics central to foundational ontology and highly debated in the literature. An introduction to these debates is out of the scope of this paper, we refer the interested reader to Sider (2001) for an interesting presentation.

Informally, we use the term *individual* to refer to entities that cannot have instances, that is, entities that cannot be predicated of others. For example, Aristotle, the Tour Eiffel and the Mars planet are individuals. On the contrary, the term *property* denotes entities that can have instances, i.e., entities that qualify other entities, e.g. Red (the color), Person (the kind), Fiat Panda (the car type).

DOLCE-CORE provides three different options to represent properties and temporary properties. The first option consists in the introduction of an *extensional predicate* as in the standard technique for the formalization of categories and primitives. In this option, to model temporal change one uses a binary predicate with a temporal parameter as in expression $F(a, t)$; here *F* is a predicate reserved for the given property, *a* an individual to which the property applies, *t* a time and expression $F(a, t)$ states that *a* has property *F* at *t*.⁸ To be more precise, since we aim at a wide-ranging view we read formulas of this form as done in Merricks (1994): *a exists at t* and it *has* the property *F* at *t* (i.e., when *t* is/was/will be present). The change from a property to one incompatible with it (as in changing colors) is then neutrally represented by writing $F(a, t_1) \wedge G(a, t_2)$. Less neutral interpretations of formula $F(a, t)$ are possible, we will see this later. Note that with this choice one cannot explicate whether the property is related to contextual or social constructions. The idea is that predicates are reserved to model the basic conceptualization of the world that the user takes for granted. Summing up, this option is to be preferred for static and context-independent properties.

⁷Here we discuss the case of properties only. Relations are treated analogously.

⁸This solution allows to represent dynamics in the properties but it introduces a series of problems when considering roles, see Steimann (2000).

The second option consists in reifying properties, that is, in associating them to entities (here called *concepts*) that are included in the domain of quantification. In order to deal with concepts (and to relate concepts to an entity according to the properties the latter has), a possibly intensional “instance-of” relation, called *classification*, is introduced in the ontology. The idea is to use concepts to represent properties whose intensional, contextual, or dynamic aspects are deemed important: “being a student”, “being a catalyst”, “being money”. As we will see, cf. (A10), concepts in DOLCE-CORE are entities in time, they are created, can be destroyed, etc. We proceed along the lines of Masolo et al. (2004) that introduces concepts to represent *roles*⁹ in terms of relationships among different entities. The properties that are best captured in this way are *anti-rigid* (Guarino and Welty, 2004), that is, those that are not essential to any of their instances. Roles provide a clear example: one (a person) may play a role (student) for a limited time, perhaps resume it in different periods, and yet she (the person) is not ontologically altered by having that role or else. Furthermore, different entities can play exactly the same role, perhaps at the same time, and a single entity can play several roles contemporarily.

The third option relies on the notions of *individual quality*, *quality-type* and *quality-space*. Similarly to concepts, with this option one can characterize the intensional, temporal, and contextual dimensions of properties. The novelty is that in this case, as we explain below, one can model also the interconnection among the way different individuals manifest a property: the quality-types isolate properties that can be meaningfully compared (like colors, weights, smells, temperatures, etc.) and quality-spaces (*spaces* for short) provide different ways to evaluate properties.

For each property of an individual, there is a correspondent individual quality (*quality* for short) that existentially depends on the individual itself. The quality *inheres in* the individual and models the specific way in which that individual has that property. The color of my car depends on my car and it is necessarily different from the color of my phone (that depends on my phone) even though both are the same shade of red: indeed the way my car is red is different from the way my phone is red. In this view, we say that my car and my phone have the *same* color because their individual qualities *exactly resemble* each other (individual qualities of different entities are necessarily distinct).

Properties can be more or less specific, compare “being scarlet” and “being red”. In the philosophical tradition, often only the more specific properties (aka *basic properties*) are assumed to correspond to truly *ontological* properties. In some approaches (Armstrong, 1997). Johansson (2000) even general properties (universals) are counted as ontological thus including properties like “being colored”, “being shaped” and the like. The idea is that these universals need to exist in order to conceive, e.g., the functional laws of physics (Armstrong, 1997). Johansson (2000)

⁹Differently from that approach, here we do not rely on logical definitions for concepts, instead the intensional aspect is (partially) characterized by explicitly stating when concepts are different. Reviews of this topic that cover a variety of perspectives are in Steiman, (2000), Masolo et al. (2004) and Loebe (2007).

isolates these properties, which characterize what we call the quality types, in terms of *maximal incompatibility* and *maximal comparability*: (i) each entity that has a quality type F must have just one basic property that is a specification of F , and (ii) all the basic properties that are specifications of F are qualitatively comparable. Qualities that share a basic property are exactly similar, while qualities that share a non-basic property are only *inexactly* similar in the sense that they resemble each other but only up to a *degree*. In applications, a variety of degrees of resemblance are empirically determined (due to species preferences, culture, available information, adopted measurement instruments or methods etc). From an empirical, applicative, and cognitive perspective, we need to recognize that properties can be arranged in different taxonomies each motivated by particular resemblance relations whose interest is motivated by needs in some domain or application. Furthermore, sometimes properties are structured in complex ways: complex topological or geometrical relations on properties are common like in the case of the color's splinter (Gärdenfors, 2000).

The use of spaces in DOLCE as complex structures of properties is inspired by Gärdenfors (2000). In this view, it is natural to think that quality types partition the individual qualities and that each quality type is associated to one or more spaces (motivated by culture, instruments of investigation, application concerns etc.) Therefore while a quality type collects the qualities that can be compared (one can reserve a quality type for color, one for smell, one for temperature etc.), quality spaces provide a way for classifying individual qualities that are in the same quality type (a variety of color classifications are used in physics, manufacturing, fashion etc., each of these isolates a different quality space).

As clear from the examples, some spaces are motivated by applications. In particular, spaces may rely on *relative* notions of resemblance: instruments present different sensitivities and each distinguishes aspects of entities only up to some *granularity*. This fact allows to order groups of spaces according to a notion of granularity (one can even postulate the existence of a space of finest granularity that recognizes all ontologically possible distinctions). In sum, spaces provide a way to make room for “subjective” (context dependent, qualitative, etc.) points of view on qualities: a quality type for color can provide the whole color spectrum, another a rough distinction in warm-cold colors, a third may discriminate by looking at brightness only etc. Note that spaces can be combined to model complex properties, i.e., properties seen as the result of interaction of other properties (the latter are then considered more basic, in a sense). This choice is often preferred when modeling properties like velocity, density and force but, as we have seen, it can be used also to structure basic properties like color.

Finally, changes in individual qualities are explained by changes in their space locations: the fact that my phone changes color is represented by the fact that over time the individual color-quality of my phone changes location in the color-space. Since the qualities of an entity exist whenever the entity exists, this third modeling option is to be considered only for properties that are necessary to an entity. Typical examples for physical objects, beside color, are mass and shape.

13.5 Basic Categories

DOLCE-CORE takes into account only entities that exist in time called *temporal particulars*. With respect to the original DOLCE neither abstract entities nor abstract qualities are considered. Our subjective perspective on spaces and the consequent idea that spaces may be created, adopted, abandoned, etc. induces us to introduce regions in time. This is different from the original DOLCE where regions are considered to be abstract entities. However, DOLCE abstract regions can be “simulated” in DOLCE-CORE by means of regions that exist at all times, i.e. regions that are neither created nor abandoned. Following Masolo et al. (2004), a similar argument holds for *concepts* (not considered in the original DOLCE) which therefore exist in time as well. Indeed, honestly, it is yet unclear to us which of the abstract or the temporal view is more appropriate for a general (and fairly comprehensive) ontology.

DOLCE-CORE partitions *temporal-particulars* (PT_i) (thereafter simply *particulars*) into *objects* (O), *events* (E), *individual qualities* (Q), *regions* (R), *concepts* (C), and *arbitrary sums* (AS). All these categories are rigid: an entity cannot change from one category to another over time. Note that the categories O (object) and E (event) correspond to the DOLCE’s categories ED (endurant) and PD (perdurant), respectively. This change in terminology is motivated by the observations in Section 13.11.

Individual qualities are themselves partitioned into *quality types* (Q_i). To each quality type Q_i are associated one or more *spaces* (S_{ij}), to the result that individual qualities in Q_i have locations in (i.e. they are located in regions belonging to) the associated spaces S_{ij} . Since we impose that the spaces are disjoint, regions are themselves partitioned into the spaces S_{ij} . For the sake of simplicity, we here consider a unique space T for (regions of) times. All these statements are enforced in the system by logical axioms although we do not report them here.

13.6 Parthood

Although mereology, the theory of parthood, is nowadays mostly used in modeling the spatial or spatio-temporal domain, the theory is not limited to this; it applies equally well to entities that are only in time (like, for instance, word meanings, beliefs, desires, societies) or that are neither in space nor in time. Indeed, mereology was introduced by Lesniewski (1991) as an alternative to set theory (the latter is based on cognitively unsatisfactory notions like the empty set and the distinction between urelements and sets) while maintaining the same level of generality.

Since the usefulness of a foundational ontology relies on the balance between ontological constraints and freedom, it is advisable to start with an ontologically weak theory and add (carefully and systematically) all the needed constraints. This approach suggests that the weak ontological commitment of mereology (at least when compared to set-theory) together with its cognitive acceptability is providing an acceptable basis for DOLCE-CORE.

Some authors tend to identify mereology with spatio-temporal inclusion. If it is true that spatio-temporally extended entities that are one part of the other are also spatio-temporally coincident, the vice versa does not hold in general: it is possible to maintain in mereology that the clay constituting a statue and the statue itself are not one part of the other although they are spatio-temporally coincident entities (see Rea (1996) for a discussion of this topic). In particular, DOLCE-CORE carefully distinguishes spatio-temporal inclusion from formal parthood. Indeed, DOLCE-CORE adopts the axioms of extensional mereology, namely (A1)–(A4), and apply them to all entities in the domain. Note also that the existence of the sum of two entities is not generally enforced: this choice depends on the entities one has in the domain (which, in turn, depends on the use one wants to do of the ontology). In short, the user of the ontology is free to impose existence of sum as a further constraint, to accept it only restricted to some categories or even to reject it in general.

In DOLCE-CORE, parthood (P) is defined on the whole domain, $P(x, y)$ stands for “ x is part of y ”.

- D1** $O(x, y) \triangleq \exists z(P(z, x) \wedge P(z, y))$ (*x and y overlap*)
D2 $SUM(z, x, y) \triangleq \forall w(O(w, z) \leftrightarrow (O(w, x) \vee O(w, y)))$ (*z is the mereological sum of x and y*)
A1 $P(x, x)$ (*reflexivity*)
A2 $P(x, y) \wedge P(y, z) \rightarrow P(x, z)$ (*transitivity*)
A3 $P(x, y) \wedge P(y, x) \rightarrow x = y$ (*antisymmetry*)
A4 $\neg P(x, y) \rightarrow \exists z(P(z, x) \wedge \neg O(z, y))$ (*extensionality*)
A5 If ϕ is O, E, Q_i , S_{jk} , or C,: (*dissectivity*)
 $\phi(y) \wedge P(x, y) \rightarrow \phi(x)$
A6 If ϕ is O, E, Q_i , S_{jk} , or C: (*additivity*)
 $\phi(x) \wedge \phi(y) \wedge SUM(z, x, y) \rightarrow \phi(z)$

Axiom (A4) states that if x is not part of y , then there is at least a part of x that does not overlap y . Axiom (A5) states that elements of a category have only parts that belong to the same category (closure under parthood) while axiom (A6) states that summing elements of a category one obtains an element in the same category (closure under sum).

13.7 Time

An ontology that aims at wide applicability has to model time. Furthermore, among the entities in the domain of quantification, it has to distinguish those that exist *in time* and, for these, *when* they exist. The expression $PRE(x, t)$ in DOLCE-CORE stands for “ x is present at t ” where the second argument t is a time. Times form the special class T in the ontology but this is done with commitment to neither a specific kind of times (points vs. intervals) nor a specific structure of time (dense vs. discrete,

linear vs. branching, etc.). Also, because of our weak reading of the existential quantifier, times in DOLCE-CORE may be considered full-fledged ontological entities or simply conceptual entities. The latter case is illustrated, for instance, by the construction of times from events (Kamp, 1979), a construction that can be adopted in this ontology.

The structure of DOLCE-CORE makes times and PRE compatible with both a *substantialist* position (the Newtonian view that time is absolute, a container-like manifold) and a *relativist* position (the Leibnizian view that time is conceptually constructed from events). This lack of commitment is important since there are alternative ways to model times depending on the application interests and the temporal information one wants to represent. E.g., being present at a time can be reduced to being simultaneous with (being before, being after) some other entity (Simons, 1991), or to being located at one specific region in a temporal quality space (Masolo et al., 2004). In addition, note that we are not distinguishing different ways of being in time like “existing in time” vs. “occurring in time” (related to the distinction objects vs. events discussed in Section 13.11) or “being wholly present” vs. “being partially present” (see the distinction endurants vs. perdurants in Section 13.8).

In short, $\text{PRE}(x, t)$ is a minimal representation device that is needed just to identify the entities that are in time and that is neutral with respect to the different ontological commitments on time, existence of events, temporal relations, theories of properties, etc. Due to the limited space, we do not enter into further details on time (for instance, on the additional constraints one can add to commit to one or the other position). A few axioms of general interest are¹⁰: x is present at t only if t is a time (A7) and being present is disjunctive and additive over t (A8) and (A9). Note that (A8) characterizes $\text{PRE}(x, t)$ as “ x is present at the whole t ”, i.e. it is not possible to find a sub-time of t at which x is not present.

A7 $\text{PRE}(x, t) \rightarrow \text{T}(t)$

A8 $\text{PRE}(x, t) \wedge \text{P}(t', t) \rightarrow \text{PRE}(x, t')$

A9 $\text{PRE}(x, t') \wedge \text{PRE}(x, t'') \wedge \text{SUM}(t, t', t'') \rightarrow \text{PRE}(x, t)$

Since, as stated in Section 13.5, in this paper we limit our discussion to the DOLCE-CORE fragment, it happens that all the entities here considered exist in time, that is

A10 $\text{PT}_t(x) \rightarrow \exists t(\text{PRE}(x, t))$

(all entities of the DOLCE-CORE fragment exist in time)

Of course, not all the entities in the general DOLCE ontology exist in time. In this case, it is enough to consider a new “top” category that includes both temporal and abstract particulars.

¹⁰Given the assumption of having just one time-space T , the constraint $\text{T}(t) \rightarrow \text{PRE}(t, t)$ can be added without any additional restriction (see also axiom (A45)).

13.8 Temporary Parthood

DOLCE-CORE adopts a temporary extensional mereology, also denoted by P , which is based on axioms (A12)-(A15), i.e., those of extensional mereology (Section 13.6) enriched with an extra temporal argument. Further mereological aspects are also enforced (see below the constraints for time regular relations). Expression $P(x, y, t)$ stands for “ x is part of y at time t .”

- D3** $O(x, y, t) \triangleq \exists z(P(z, x, t) \wedge P(z, y, t))$ (x and y overlap at time t)
A11 $P(x, y, t) \rightarrow PRE(x, t) \wedge PRE(y, t)$ (parthood implies being present)
A12 $PRE(x, t) \rightarrow P(x, x, t)$ (temporary reflexivity)
A13 $P(x, y, t) \wedge P(y, z, t) \rightarrow P(x, z, t)$ (temporary transitivity)
A14 $PRE(x, t) \wedge PRE(y, t) \wedge \neg P(x, y, t) \rightarrow \exists z(P(z, x, t) \wedge \neg O(z, y, t))$ (temporary extensionality)
A15 If ϕ is O, E, Q_i, S_{jk} or C : $\phi(y) \wedge P(x, y, t) \rightarrow \phi(x)$ (temporary dissectivity)

For standard parthood we stated axiom (A3) so that entities indistinguishable with respect to parthood are identical. This claim does not hold when temporary parthood is involved. Temporal coincidence (D4) provides a suitable form of identification: two entities x and y that are temporary coincident at time t , formally $CC(x, y, t)$, are indistinguishable relatively to time t but can still be different in general.¹¹ If $CC(x, y, t)$, then all the properties of x at t are also properties of y at t and vice versa.

For properties that are formalized via concepts or qualities, the constraint is explicitly introduced by the *substitutivity* axioms. In the case of the primitive relations of *classification* and *location* (that we will introduce later) an axiom of the form (SB) (given below) is enough, while in the case of the *inherence* relation, axioms (A25) and (A26) do the work. Note however that from these axioms no constraint follows on properties of x and y at a time different from t nor on properties represented by means of additional predicates introduced by the user.

Axiom (A16) states that, for entities in time, parthood simpliciter can be defined on the basis of temporary parthood. The opposite is true only if one commits to the existence of temporal parts (at every time of existence), an option compatible with both DOLCE and DOLCE-CORE but that is not enforced. This means that the axioms for temporary parthood are compatible with both the endurantist and perdurantist views of persistence through time. Assuming that x and y persist through time, endurantists read the formula $P(x, y, t)$ as “ x and y are both *wholly* present at t and x is part of y ”, while perdurantists read that formula as “the temporal part of x at t is part of the temporal part of y at t ”. Therefore, perdurantists need to assume the existence of x and y as well as that of the temporal parts of x and y .

¹¹From a perdurantist perspective (see Section 13.11) where entities are considered as four-dimensional “worms”, this simply means that two possibly different four-dimensional worms (x and y) have the same temporal slice at t .

- D4** $CC(x, y, t) \triangleq P(x, y, t) \wedge P(y, x, t)$ (*x, y coincide at t*)
D5 $CP(x, y) \triangleq \exists t(PRE(x, t)) \wedge \forall t(PRE(x, t) \rightarrow P(x, y, t))$ (*x is a constant part of y*)
A16 $\exists t(PRE(x, t)) \rightarrow (CP(x, y) \leftrightarrow P(x, y))$
(for entities in time, “constant part” and “parthood” are equivalent)

Temporary parthood presents three main novelties with respect to the corresponding relationship of DOLCE: (i) it is defined on all the particulars that are in time; (ii) the existence of sums is not guaranteed; (iii) (A16) is new (in DOLCE it was considered as a possible extension).

Let us say that a relation R is *time regular* whenever it satisfies the following:

- (DS) $R(x, y, t) \wedge P(t', t) \rightarrow R(x, y, t')$ (*dissectivity*)
(AD) $R(x, y, t') \wedge R(x, y, t'') \wedge SUM(t, t', t'') \rightarrow R(x, y, t)$ (*additivity*)
(SB) $R(x, y, t) \wedge CC(x', x, t) \wedge CC(y', y, t) \rightarrow R(x', y', t)$ (*substitutivity*)

We can rephrase these properties as follows: if the relation holds at a time, it holds at any sub-time; if the relation holds at two times, then it holds also at the time spanning the two; if a relation holds at t for two entities, then it holds for any two entities temporally coincident at t with them. These properties are collected here since they characterize several relations in DOLCE-CORE. In particular, we conclude our partial presentation of temporary parthood by stating that this relation is time regular, that is, it satisfies all the above constraints.

13.9 Concepts

The formalization of properties as extensional predicates (the first option of Section 13.4) is straightforward and requires no new formal element. Instead, the second option we considered involves two notions which are not in the original version of DOLCE: the category of concepts C and the relation of classification CF . $CF(x, y, t)$ stands for “ x classifies y at time t ” and is characterized in DOLCE-CORE as a time regular relation that satisfies

- A17** $CF(x, y, t) \rightarrow C(x)$
A18 $CF(x, y, t) \rightarrow PRE(y, t)$

In addition we require that concepts are mereologically constant, i.e., with respect to parthood they do not change over time:

- A19** $C(x) \wedge PRE(x, t) \wedge PRE(x, t') \rightarrow \forall y(P(y, x, t) \leftrightarrow P(y, x, t'))$

13.10 Qualities and Locations

The third option to formally represent properties is via individual qualities.

Each individual quality, say “the color of my car” or “the weight of John”, and its host are in a special relationship called *inheritance*. Formally, it is expressed

by expressions of form $I(x, y)$, whose intended reading is “the individual quality x inheres in the entity y ”.¹² This relationship binds a specific bearer as shown by (A21) while (A22) says that each quality existentially depends on the entity that bears it; in the previous examples the bearers are my car and John, respectively. Furthermore, from axiom (A23) qualities exist during the whole life of their bearers.¹³

We anticipated that individual qualities are grouped into quality types, say Q_i is the color-quality type, Q_j the weight-quality type etc. These constraints are simple and we do not report them explicitly except for axiom (A24) according to which an entity can have at most one individual quality for each specific quality type. Axioms (A25) and (A26) say that if two particulars coincide at t then they need to have qualities of the same type and these qualities also coincide at t . In other terms, entities coincident at t must have qualities that are indistinguishable at t . Axiom (A27) says that the sum of qualities of the same type that inhere in two objects inheres in the sum of the objects (provided these sums exist).

$$\mathbf{A20} \quad I(x, y) \rightarrow Q(x)$$

$$\mathbf{A21} \quad I(x, y) \wedge I(x, y') \rightarrow y = y'$$

$$\mathbf{A22} \quad Q(x) \rightarrow \exists y(I(x, y))$$

$$\mathbf{A23} \quad I(x, y) \rightarrow \forall t(\text{PRE}(x, t) \leftrightarrow \text{PRE}(y, t))$$

$$\mathbf{A24} \quad I(x, y) \wedge I(x', y) \wedge Q_i(x) \wedge Q_i(x') \rightarrow x = x'$$

$$\mathbf{A25} \quad \text{CC}(x, y, t) \rightarrow (\exists z(I(z, x) \wedge Q_i(z)) \leftrightarrow \exists z'(I(z', y) \wedge Q_i(z')))$$

$$\mathbf{A26} \quad \text{CC}(x, y, t) \wedge I(z, x) \wedge I(z', y) \wedge Q_i(z) \wedge Q_i(z') \rightarrow \text{CC}(z, z', t)$$

$$\mathbf{A27} \quad I(x, y) \wedge I(v, w) \wedge Q_i(x) \wedge Q_i(v) \wedge \text{SUM}(z, x, v) \wedge \text{SUM}(s, y, w) \rightarrow I(z, s)$$

Note that we do not force a schema of form

$$\text{Rejected} \quad I(x, y) \wedge Q_i(x) \wedge P(y', y) \rightarrow \exists x'(I(x', y') \wedge Q_i(x') \wedge P(x', x))$$

because this would prevent properties that inhere in complex objects only, e.g., emergent properties like functionalities of assembled artifacts (when not reducible to functionalities of the components).

The *location* relation, L , provides the link between qualities and spaces. First, we require regions (and in particular spaces) not to change during the time they exist (A28). Then, we write $L(x, y, t)$ to mean “at time t , region x is the location of the individual quality y ” as enforced (in part) by axioms (A30) and (A31).¹⁴ Each individual quality of type Q_i must be located at least in one of the available spaces S_{ij} associated to it (axioms (A34) and (A35)). The location in a single space is unique

¹²In the original version of DOLCE this relation is called *quality* and written qt .

¹³For those familiar with trope theory (Campbell, 1990), qualities can be seen as sums of tropes. Indeed, one can interpret a trope substitution as a change of quality location. The position adopted in DOLCE-CORE is compatible with trope theory without committing to the view that change corresponds to trope substitution.

¹⁴In the original version of DOLCE this relation is called *quale* and written ql . In DOLCE there was also a distinction between the *immediate* quale (a non temporary relation) and the *temporary* quale. Here we use one temporary relation only and assume that the temporal qualities of an event e at t correspond to the temporal qualities of the maximal part of e spanning t .

(A36) and a quality that has a location in a space needs to have some location in that space during its whole life (A37). (A38) says that two qualities coincident at t are also indistinguishable with respect to their locations. Together with (A25) and (A26), this axiom formalizes the substitutivity of temporary properties represented by qualities: two entities that coincide at t are (at t) indistinguishable with respect to their qualities.

Axioms (A32) and (A33) characterize the fact that the location of an individual quality at t is the mereological sum of all the locations the quality has *during* t , i.e. at all the sub-times of t . Note that if a is the region corresponding to a property value of $1kg$ and b corresponds to a property value of $2kg$, then the sum of a and b is the region including just the two mentioned and is distinguished from the region corresponding to the property value of $3kg$. The sum of locations must not be confused with the sum of property values since, in general, the latter strictly depends on the space structure while the first does not. Therefore, for instance, if t is the sum of t_1 and t_2 , and $L(1kg, x, t_1)$ and $L(1kg, x, t_2)$, then at t , x is still located a $1kg$ and not at $2kg$.

$$\mathbf{A28} \quad R(x) \wedge \text{PRE}(x, t) \wedge \text{PRE}(x, t') \rightarrow \forall y(\text{P}(y, x, t) \leftrightarrow \text{P}(y, x, t'))$$

$$\mathbf{A29} \quad S_{ij}(x) \wedge S_{ij}(y) \wedge \text{PRE}(x, t) \rightarrow \text{PRE}(y, t)$$

$$\mathbf{A30} \quad L(x, y, t) \rightarrow R(x) \wedge Q(y)$$

$$\mathbf{A31} \quad L(x, y, t) \rightarrow \text{PRE}(y, t)$$

$$\mathbf{A32} \quad L(x, y, t) \wedge \text{P}(t', t) \wedge L(x', y, t') \wedge S_{ij}(x) \wedge S_{ij}(x') \rightarrow \\ \forall t''(\text{PRE}(x, t'') \rightarrow \text{P}(x', x, t''))$$

$$\mathbf{A33} \quad L(x', y, t') \wedge L(x'', y, t'') \wedge \text{SUM}(t, t', t'') \wedge \text{SUM}(x, x', x'') \wedge S_{ij}(x') \wedge S_{ij}(x'') \rightarrow \\ L(x, y, t)$$

$$\mathbf{A34} \quad L(x, y, t) \wedge Q_i(y) \rightarrow \bigvee_j S_{ij}(x)$$

$$\mathbf{A35} \quad Q(y) \wedge \text{PRE}(y, t) \rightarrow \exists x(L(x, y, t))$$

$$\mathbf{A36} \quad L(x, y, t) \wedge L(x', y, t) \wedge S_{jk}(x) \wedge S_{jk}(x') \rightarrow x = x'$$

$$\mathbf{A37} \quad L(x, y, t) \wedge \text{PRE}(y, t') \wedge S_{jk}(x) \rightarrow \exists x'(L(x', y, t') \wedge S_{jk}(x'))$$

$$\mathbf{A38} \quad L(x, y, t) \wedge \text{CC}(x', x, t) \wedge \text{CC}(y', y, t) \rightarrow L(x', y', t) \quad (\text{L-substitutivity})$$

The next formula is not an axiom since not all properties are dissective (see the previous example using weights)

$$\underline{\text{Rejected}} \quad L(x, y, t) \wedge \text{P}(t', t) \rightarrow L(x, y, t')$$

Additivity is also non-valid for L: it does not hold for properties like mass.

$$\underline{\text{Rejected}} \quad L(x, y', t) \wedge L(x, y'', t) \wedge \text{SUM}(y, y', y'') \rightarrow L(x, y, t)$$

13.11 Objects and Events

We all experience a tendency to distinguish *what changes* from the *changing* itself. A lively and long discussion on the ontological status of events and on what distinguishes them from objects has taken place especially in the philosophy of language

(Casati and Varzi, 1996). There are formal and applicative advantages in accepting events, e.g., one can (i) quantify over actions, (ii) predicate on causality, and (iii) avoid reductionist views.

The original DOLCE formulated the object vs. event distinction in terms of the enduring vs. perdurant partition by identifying objects with endurants and events with perdurants. This choice reflects the position of several philosophers and is based on the observation that, say, the “life of John” is only *partially* present at each time at which it exists (it has distinct temporal parts at each time at which it exists) while “John” is *wholly* present whenever it exists (it does not depend on the existence of temporal parts). However, with this position classical perdurantism would not be able to embrace the object vs. event distinction for the simple reason that perdurantists accept only perdurant entities. In addition, in a strict reading of perdurantism, all particulars must be spatio-temporally extended and two distinct entities cannot have exactly the same spatio-temporal location. Thus, since “John” and “the life of John” have exactly the same spatio-temporal location, perdurantists would be forced to identify them. This shows that the previous identification, although motivated by some aspects of the theories, is perhaps too naïve and a different (and more general) foundation of the distinction between objects and events should be sought.

Hacker (1982) proposes to characterize the distinction on the fact that events are *primarily* in (directly related to) time while material, and more generally physical, objects are *primarily* in (directly related to) space. Indeed,

- the properties (and qualities) that apply to material objects are different from those that apply to events. Typically, material objects have weight, size, shape, texture, color etc. and they are related by specific spatial relationships like congruence. Events, on the other hand, can be sudden, brief or prolonged, fast or slow, etc. and can occur before, after, simultaneously with other events. Moreover, relations like causation seem to be strictly linked to events and not to objects.
- Space plays a role in the identification of material objects, time in that of events. Material objects that are simultaneously located at different places are different and events that have different temporal locations are different (Zemach, 1970).
- The unity criteria of objects is primarily spatial, while the one of event is primarily temporal (Hacker, 1982).

This division extends to non-material objects as well since these are also characterized by non-temporal properties and specific individuation and unity criteria (space is the realm of these criteria for material objects, other objects rely on other dimensions, all distinct from time). In short, what differentiates events from (material or immaterial) objects is the special connection to time and temporal relations.

Of course, even though events are primarily in time while objects are primarily in other dimensions, there are strong interrelationships between them. Several authors (Simons, 1991; Hacker, 1982) claim that events are not possible without

objects and vice versa. Since from the representation perspective there seems to be no real advantage in committing to a reductionist view (either choosing that events are the truly basic entities or, on the contrary, attributing to objects this role), the preferred option is to consider both categories of events and of objects as primary categories and to highlight their relationships: events need participants (objects) and objects need lives (events). By means of the relationship between objects and events (aka *participation*), it is possible to say that an object a exists at a certain time t “if and because” its life exists at t (Simons, 2000), i.e. it is the life of a that is the truth-maker for the proposition “ a exists at t ”. On the other hand, events are related to space only indirectly via the material objects participating in them.

DOLCE-CORE characterizes the distinction between objects and events (two basic categories in the ontology structure) following this latter approach. However, by (A10), qualities, concepts, and regions are in time too and, intuitively, their participation to events (like their creation or destruction) seems plausible. In this perspective, qualities, concepts and regions could be considered as subcategories of O (in this view objects are not necessarily extended in space). Here we do not commit to this position and therefore we maintain our initial assumption where qualities, concepts, and regions are disjoint from objects.

Participation is taken to be a *time regular* relation defined between objects and events: $PC(x, y, t)$ stands for “the entity x participates in the event y at t ”. Axioms (A40) and (A41) capture the mutual existential dependence between events and objects. Axioms (A42) and (A43) make explicit the fact that participation does not rely on unity criteria for objects or for events (Simons, 1987). This simply means that the participation relation is not bound by these unity criteria: an object does not participate to an event as a whole (since also its parts participate to it) as well as an event does not have participants because of some special unity property (since all the events, of which it is part, have those participants too). Participation, of course, can be used to define more specific relations that take into account unity criteria. Since these criteria often depend on the purposes for which one wants to use the ontology, they are not discussed here.

$$\mathbf{A39} \quad PC(x, y, t) \rightarrow O(x) \wedge E(y)$$

$$\mathbf{A40} \quad E(x) \wedge PRE(x, t) \rightarrow \exists y(PC(y, x, t))$$

$$\mathbf{A41} \quad O(x) \wedge PRE(x, t) \rightarrow \exists y(PC(x, y, t))$$

$$\mathbf{A42} \quad PC(x, y, t) \wedge P(y, y', t) \wedge E(y') \rightarrow PC(x, y', t)$$

$$\mathbf{A43} \quad PC(x, y, t) \wedge P(x', x, t) \rightarrow PC(x', y, t)$$

We now clarify how DOLCE-CORE manages to formalize events and objects as entities with different qualities, and how it represents “being *primarily* in time (space)”. Axiom (A44) makes explicit the fact that quality types directly connected to events cannot be directly related to objects and vice versa.

$$\mathbf{A44} \quad I(x, y) \wedge Q_i(x) \wedge E(y) \wedge I(z, v) \wedge Q_j(z) \wedge O(v) \rightarrow \neg Q_j(x) \wedge \neg Q_i(z)$$

The exact quality types that apply to objects and events depend on the modeling interests of the user. Nonetheless, as motivated earlier, qualities that apply to events are strictly connected to time (fast vs. slow, sudden vs. prolonged, etc.).

Regarding the property of “being primarily in time”, we introduce the quality type “being time-located”.¹⁵ Let us use TQ for this quality type for time and let us make the simplifying assumption that there is just one space for the time individual qualities; as seen in Section 13.5, we call it T. DOLCE-CORE (as well as DOLCE) distinguishes *direct* qualities, i.e., properties that can be predicated of x because it has a corresponding individual quality, from *indirect* qualities, i.e., properties of x that are inherited from the relations x has with other entities. For instance, following Simons (2000), events have a direct temporal location, while objects are located in time just because they participate to events (e.g. their own lives). Analogously, material objects have a direct spatial location, while events are indirectly located in space through the spatial location of their participants.

(A45) makes explicit the temporal nature of the parameter t in the location relation. (A46) says that for events “being in time” reduces to having a time-quality located in time, which, together with (A10) and (A44), guarantees that all and only the events have a time-quality. These axioms together with (A41), show that objects are in time because of their participation in events.

$$\mathbf{A45} \quad L(x, y, t) \wedge \text{TQ}(y) \rightarrow x = t$$

$$\mathbf{A46} \quad E(x) \wedge \text{PRE}(x, t) \leftrightarrow \exists y(\text{TQ}(y) \wedge I(y, x) \wedge L(t, y, t))$$

Note that if we define the spatial location of events via the location of their participants, and the life of an object as the minimal event in which it (maximally) participates, we obtain that an object spatio-temporally coincides with its life. The distinction between participation, temporary parthood, and spatial inclusion ensures that these two entities, although spatio-temporally coincident, are not identified.

As stated before, in DOLCE-CORE qualities cannot participate in events. This holds in particular for the qualities of objects even though in this case qualities are related to time by axiom (A23) through the objects they inhere in (which necessarily exist by axiom (A22)). Time is (at least) an “indirect” quality of qualities of objects. However one could follow the opposite intuition that qualities can participate in events, assuming that only qualities are the “true” participants in events. In this perspective, objects would participate only indirectly *because of* the qualities they have.

Several important questions have been left out of this paper. We hope, nonetheless, that the approach adopted in building first DOLCE and then DOLCE-CORE stands out, that the foundational choices we have made can be appreciated for the careful analysis they rely upon, and that the general methodology we apply here has been fairly illustrated.

¹⁵Analogously, the ontology comprises the quality type “being space-located” which is not presented here.

References

- Armstrong, D.M. 1989. *Universals: An opinionated introduction*. Boulder, CO: Westview Press.
- Armstrong, D.M. 1997. *A world of states of affairs* (Cambridge Studies in Philosophy). Cambridge, MA: Cambridge University Press.
- Borgo, S., and C. Masolo. 2009. Foundational choices in DOLCE. In *Handbook on ontologies* (2nd Edition), eds. S. Staab, and R. Studer, 361–382. Berlin: Springer.
- Campbell, K. 1990. *Abstract particulars*. Oxford: Basil Blackwell.
- Casati, R., and A.C. Varzi. eds. 1996. *Events*. Aldershot: Dartmund.
- Gangemi, A., N. Guarino, C. Masolo, and A. Oltramari. 2003. Sweetening wordnet with dolce. *AI Magazine* 24(3):13–24.
- Gärdenfors, P. 2000. *Conceptual spaces: The geometry of thought*. Cambridge, MA: MIT Press.
- Guarino, N., and C. Welty. 2004. An overview of ontoclean. In *Handbook of Ontologies*, eds. S. Staab, and R. Studer, 151–172. Berlin: Springer.
- Hacker, P.M.S. 1982. Events and objects in space and time. *Mind* 91:1–19.
- Heil, J. 2005. *From an ontological point of view*. Oxford: Oxford University Press.
- Johansson, I. 2000. Determinables as universals. *The Monist* 83(1):101–121.
- Kamp, H. 1979. Events, istants and temporal reference. In *Semantics from different points of view*, eds. R. Bäuerle, U. Egli, and A. von Stechow, 376–417. Berlin: Springer.
- Lesniewski, S. 1991. *Collected works*. Dordrecht: Kluwer.
- Loebe, F. 2007. Abstract vs. social roles – towards a general theoretical account of roles. *Applied Ontology* 2(2):127–258.
- Loux, M.J., ed. 1976. *Universals and particulars: Readings in ontology*. London: University of Notre Dame Press, Copia.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. Wonderweb deliverable d18. Technical report, CNR.
- Masolo, C., L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. 2004. Social roles and their descriptions. In Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning, Whistler, Canada.
- Mellor, D.H., and A. Oliver. eds. 1997. *Properties*. Oxford: Oxford University Press.
- Merricks, T. 1994. Endurance and indiscernibility. *Journal of Philosophy* 91(4):165–184.
- Prevot, L., S. Borgo, and A. Oltramari. 2005. Interfacing ontologies and lexical resources. In *Ontologies and Lexical Resources: IJCNLP-05 Workshop*, 1–12.
- Rea, M. ed. 1996. *Material constitution*. Lanham, MD: Rowman and Littlefield Publishers.
- Sider, T. 2001. *Four-dimensionalism. An ontology of persistence and time*. Oxford: Clarendon Press.
- Simons, P. 1987. *Parts: A study in ontology*. Oxford: Clarendon Press.
- Simons, P. 1991. On being spread out in time: Temporal parts and the problem of change. In *Existence and explanation*, ed. W. Spohn et al., 131–147. Kluwer Academic Publishers.
- Simons, P. 2000. How to exist at a time when you have no temporal parts. *The Monist* 83(3): 419–436.
- Steimann, F. 2000. On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering* 35:83–106.
- Zemach, E.M. 1970. Four ontologies. *Journal of Philosophy* 67(8):231–247.

Chapter 14

General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling

Heinrich Herre

14.1 Introduction

Research in ontology has in recent years become widespread in the field of information systems, in distinct areas of sciences, in business, in economy, and in industry. The importance of ontologies is increasingly recognized in fields diverse as in e-commerce, semantic web, enterprise, information integration, qualitative modelling of physical systems, natural language processing, knowledge engineering, and databases. Ontologies provide formal specifications and harmonized definitions of concepts used to represent knowledge of specific domains. An ontology supplies a unifying framework for communication and establishes the basis of the knowledge about a specific domain.

The term *ontology* has two meanings, it denotes, on the one hand, a research area, on the other hand, a system of organized knowledge. A system of knowledge may exhibit various degrees of formality; in the strongest sense it is an axiomatized and formally represented theory. which is denoted throughout this paper by the term *axiomatized ontology*.

We use the term *formal ontology* to name an area of research which is becoming a science similar as formal or mathematical logic. *Formal ontology* is an evolving science which is concerned with the systematic development of axiomatic theories describing forms, modes, and views of being of the world at different levels of abstraction and granularity. Formal ontology combines the methods of mathematical logic with principles of philosophy, but also with the methods of artificial intelligence and linguistics. At the most general level of abstraction, formal ontology is concerned with those categories that apply to every area of the world.

The application of formal ontology to domains at different levels of generality yields knowledge systems which are called, according to the level of abstraction, *Top Level Ontologies* or *Foundational Ontologies*, *Core Domain* or *Domain Ontologies*.

H. Herre (✉)

Research Group Onto-Med, IMISE, University Leipzig, Leipzig, Germany
e-mail: heinrich.herre@imise.uni-leipzig.de

Contribution for the TAO-Volume

Top level or foundational ontologies apply to every area of the world, in contrast to the various *Generic, Domain Core* or *Domain Ontologies*, which are associated to more restricted fields of interest. A foundational ontology can serve as a unifying framework for representation and integration of knowledge and may support the communication and harmonisation of conceptual systems.

The current paper presents an overview about the current stage of the foundational ontology GFO.¹ GFO (General Formal Ontology) is a component of ISFO (Integrated System of Foundational Ontologies), and ISFO is intended to be a part of an Integrated Framework for Development and Application of Ontologies (IFDAO) whose predecessor was the GOL project that was launched in 1999 at the University of Leipzig. GFO is a foundational ontology integrating objects and processes. It is being developed, after a deep revision of the original GOL-project, by the research group Onto-Med (Ontologies in Medicine)² at the University of Leipzig. GFO exhibits a three-layered meta-ontological architecture consisting of an abstract top-level, an abstract core level, and a basic level. Further unique selling propositions of GFO are the following:

- Includes objects (3D objects) as well as processes (4D entities) and both are integrated into one coherent framework,
- GFO presents a multi-categorical approach by admitting universals, concepts, and symbol structures and their interrelations,
- includes levels of reality,
- is designed to support interoperability by principles of ontological mapping and reduction,
- it is presented by a set of formal axioms which might be added by meta-logical links to domain specific ontologies,³
- GFO is intended to be the basis for a novel theory of ontological modelling which combines declarative specifications with algorithmic procedures,
- it contains several novel ontological modules, in particular, a module for functions and a module for roles, and
- GFO is designed for applications, firstly in medical, biological, and biomedical areas, but also in the fields of economics and sociology.

We envision GFO to be a foundational ontology which is expressive enough to contain several other foundational ontologies as special cases. But, GFO is not intended to be the ultimate result of a foundational ontology; one may doubt whether a final and uniquely determined top level ontology can ever be achieved. For this reason, GFO is merely a component of the evolutionary system ISFO, which leaves

¹A more detailed exposition of GFO is presented in Herre et al. (2006b).

²<http://www.onto-med.de>

³The development of axiomatic systems for GFO is work in progress and will be published as Part II of the General Formal Ontology.

room for modifications, revisions, adaptations that are triggered by the historical state of our knowledge, and the applications in nature and society.

Foundational ontologies may differ with respect to their basic categories and relations (i.e. their vocabulary), with respect to the set of axioms formulated about their vocabulary or with respect to both the basic vocabulary and the axioms. If two ontologies have the same basic categories and relations, then the question arises which axioms should be included in the axiomatization. Here, we admit various different axiomatizations. The investigation of a system of axioms with respect to its possible consistent extensions and of other meta-logical properties is an interesting research topic of its own. It is our opinion that different views of the world can be sustained, though over time we expect that the number will be reduced to a few such views, mainly based on utility. According to our pluralistic approach ISFO is intended to be an integrated and evolutionary system of foundational ontologies. These ontologies are compared and interrelated using methods of translation and interpretation.

Domain-specific ontologies exhibit a much richer diversity than foundational ontologies. The notion of a domain specific ontology admits various interpretations. From a practical point of view, domain specific ontologies include, among others, terminologies, glossaries, thesauri and nomenclatures which are associated to specific domains. A terminology, for example, is a list of terms referring to concepts in a particular domain. Such a terminology is usually a result of consensus and agreement among the domain's experts. Hence, a terminology tends to converge to a unified and common basic vocabulary which is influenced and determined by utility and usage. A domain specific ontology in the stronger sense, i.e. understood as an axiomatized ontology, normally is based on a terminology. There are usually various axiomatized ontologies that may be built on a terminology.⁴

Several groups are tackling the development of top-level ontologies or certain aspects of top-level ontologies. The following approaches are fairly developed, and they are used, in part, as a source for our considerations. Nicola Guarino, an early proponent of the use of ontologies in the field of knowledge-based systems, is involved in the construction of DOLCE (Masolo et al., 2003). Further, two other ontologies⁵ are presented in Masolo et al. (2003), following the idea of an ontology library in the WonderWeb project. DOLCE itself is presented as a hierarchy of categories and several relationships between them. The description is fairly extensive and an axiomatization is contained therein as well.

⁴These axiomatized ontologies of a domain are influenced by the assumed views and the classification principles from which different conceptualizations can be derived. Furthermore, there is no sufficiently founded criterion to establish the equivalence of two ontologies. Hence, the orthogonality criterion, as expounded in Smith et al. (2007), must be rejected.

⁵One of these ontologies, called BFO (Basic Formal Ontology) (Grenon, 2003), has its source in the GOL- project which started in 1999 as a common project of the department of formal concepts (Institute for Computer Science) and the Institute for Medical Informatics of the University of Leipzig. GOL was the scientific basis for a research programme related to a Wolfgang-Paul Prize advertised in 2001. Since June 2002 GFO and BFO were independently developed.

John Sowa (2000) presents and extensively discusses a top-level ontology in the form of a polytree within a comprehensive book on knowledge representation issues, i.e. it is not a pure work introducing a top-level ontology. Sowa's ontology is based on ideas of the philosopher *Charles Sanders Peirce*. The Standard Upper Merged Ontology (SUMO) is an effort of the P1600.1 Standard Upper Ontology Working Group at IEEE (SUO, 2004). Several draft proposals have been made, one of the more developed suggestions of which is SUMO. SUMO adopts a polytree architecture of categories, in which there are cases of multiple super-categories, for example, Group is a subcategory of both Collection and Agent. Its development may have contributed to the multiplicative approach, as SUMO originates from a merge of several top-level ontologies (cf. Niles and Pease, 2001), including one of Russell and Norvig (1995), one of John Sowa (2000), as well as several others. Another relevant ontology is a 4-dimensional ontology developed by West et al. (2003). Roberto Poli contributes an additional important account of ontology in the field of computer science (Poli, 2001). In particular, Poli presents a theory of ontological levels (cf (Poli, 2002, Poli, 2001) that is acknowledged and adopted in GFO. A further important contribution to top-level ontologies is the approach by Guizzardi and Wagner (2004a, b) which is aimed at the development of a Unified Foundational Ontology (UFO). This ontology integrates several aspects of DOLCE and GFO.

The current paper is an exposition of main ideas of GFO and draws on the report (Herre et al., 2006b). In Section 14.2 the basic assumptions and logical methods are expounded. Section 14.3 describes the meta-ontological architecture of GFO. Section 14.4 is devoted to a detailed exposition of the basic categories of individuals; this section presents the core of the current paper. In Section 14.5 the basic relations of GFO are outlined. In Section 14.6 a central idea of GFO is exposed: the object-process integration. Section 14.7 presents some ideas on principles of ontology development and ontological modelling.

14.2 Basic Assumptions and Logical Methods

In this section we summarize the basic assumptions and methods from philosophy and logic.

14.2.1 Philosophical Assumptions

Ontology is based on a particular view at the world: ontology asks what an entity is, what the essence of it is, and which mode of existence it has. We use the term "entity" for everything that exists where existence is understood in the broadest sense. In Ingarden (1964) a classification of modes of existence is discussed that is useful for a deeper understanding of entities of several kinds. According to Ingarden (1964) there are –roughly- the following modes of being: absolute, ideal, real, and intentional entities. This classification can be to some extent related to Gracia's

approach (Gracia, 1999) and to the levels of reality in the spirit of Hartmann(1964) and Poli (2001). We hold a realist view at the world and assume the existence of a real world which is independent from the subject. Appearances of this world are considered as actual realizations of the world's dispositions within a subject. There is a mutual relation between the subject and object in the sense of (Bloch, 1985) which exhibits the most important and complex inter-relationship between man and nature.

Categories are entities that are expressed by predicative terms of a formal or natural language that can be predicated of other entities. Predicative terms are linguistic expressions T which state conditions $\text{Cond}(T)$ to be satisfied by an entity. Categories are what predicative terms express, not the predicative terms themselves (Gracia, 1999). There is a close relation between categories and language, hence the analysis of the notion of a category cannot be separated from the investigation of language. The predicative term T , the expressed category C , the conditions $\text{Cond}(T)$ specified by T , and the satisfying entity e are mediated by two relations, $\text{expr}(T,C)$ and $\text{sat}(\text{Cond}(T),e)$. We stipulate that a category C is predicated of an entity e if and only if e satisfies the conditions that are associated to C . Summarizing the inter-relations between categories, conditions, and predicative terms we hold that an entity e instantiates the category C with respect to T if and only if $\text{expr}(T,C)$ and e satisfies the conditions $\text{Cond}(T)$ associated to the term T .

Individuals are entities which cannot be instantiated, hence, they cannot be predicated of other entities. There is a distinction between the property of being instantiable and of having instances. Individuals cannot be instantiated, on the other hand, there are categories without any instance. The expression "round square", for example, presents a category without any instance. Such categories are called empty, and they are an extensional sub-category of every category. For a category C we introduce the *transitive closure* of C , denoted by $\text{trcl}(C)$; this is the smallest set containing C as an element and which is closed with respect to the following condition: if $D \in \text{trcl}(C)$ and $e::D$ then $e \in \text{trcl}(C)$. The *individual base* of a category C , denoted by $\text{IndBase}(C)$, is the set of all individuals which are elements of the transitive closure $\text{trcl}(C)$.

14.2.2 Concepts, Symbols, and Universals

We distinguish at least three kinds of categories: universals, concepts, and symbol structures. We hold that any reasonable foundational ontology must include these three types of categories.⁶ *Universals* are constituents of the real world, they are associated to invariants of the spatio-temporal real world, they are something abstract that is in the things. Concepts are categories that are expressed by linguistic expressions and which are represented as meanings in someone's mind. Concepts

⁶Our approach to categories is inspired by the ideas of Jorge Gracia (1999). We consider Gracia's approach as an important contribution to the philosophical foundation of conceptual modelling.

are a result of common intentionality which is based on communication and society (Searle, 1995).⁷ *Symbols* are signs or texts that can be instantiated by tokens. There is a close relation between these three kinds of categories: a universal is captured by a concept which is individually grasped by a mental representation, and the concept and its representation is denoted by a symbol structure being an expression of a language. Texts and symbolic structures may be communicated by their instances that are physical tokens.⁸

Sets play a particular role in GFO. We hold that a set cannot be predicated of its members, but there are, of course, specifications of sets expressing categories which can be predicated of sets. Concepts have a complex structure that consists of interrelated particular parts, which are called conceptual constituents. For concepts we introduce a basic relation $\text{catp}(x,y)$ having the meaning that x is a categorial part of the concept. In the simplest case a concept can be represented as a set or aggregate of predicates or properties.

14.2.3 *The Axiomatic Method*

The axiomatic method comprises principles used for the development of formal knowledge bases and reasoning systems aiming at the foundation, systematization and formalization of a field of knowledge associated with a part or dimension of reality.

The axiomatic method deals with the specification of concepts. If knowledge of a certain domain is to be assembled in a systematic way, one can distinguish a certain small set of concepts in this field that seem to be understandable of themselves. We call the expressions in this set *primitive* or *basic*, and we use them without formally explaining their meanings through explicit definitions. Examples for primitive concepts are *identity* and *part*. New terms can be introduced by explicit definitions based on the primitive notions.

Given the basic terms, we can construct more complex sentences that can be understood as descriptions of certain formal interrelations between them. Some of these statements are chosen as *axioms*; we accept them as true without establishing their validity by means of a proof. The truth of axioms of an empirical theory may be supported by experimental data. By accepting such sentences as axioms, we assert that the interrelations described are considered to be valid in some domain and at

⁷The mental representation of a concept allows us to understand a linguistic expression. Concepts are outside of individual minds, but they are anchored, on the one hand, in individual minds by the concepts' mental representation, and on the other hand, in society as a result of communication and usage of language.

⁸The ability to generate and use symbol structures seems to be the most basic assumption for complex communication. Here, an important aspect of the ability of humans to construct symbolic structures and to identify tokens as instances of symbols. The ultimate transmission of information must use spatio-temporal tokens as bearers.

the same time we define the given notions implicitly, i.e. the meaning of the basic terms is captured and constrained by the axioms.

Axiomatic theories should be studied with respect to meta-theoretical properties. It is important that the axioms of a foundational ontology are consistent, because domain-specific and generic axioms will be built on them.⁹ Other important meta-theoretical properties are completeness, and the classification of complete extensions. If several theories are considered, their inter-relationships must be studied which will lead to questions regarding the interpretation of one theory in another, and identifying the more comprehensive or expressive theory.

14.2.4 Representation of Ontologies

An ontology must be represented and specified by expressions of a language. We assume that the general terms in these expressions denote concepts. An ontology O – understood as a formal knowledge base – is given by an explicit specification of a conceptualization (Gruber, 1993). This specification must be expressed in a formal language, and there is a variety of formal specification systems. A main distinction is drawn between logical languages with model-theoretic semantics and formalisms using graph-theoretic notations.

14.2.5 Types of Realism

The type of realism on which GFO is based is called *integrative realism*; it fundamentally differs from the realism expounded in a number of papers by Smith (2004), Smith et al. (2005), Smith (2006), Smith and Ceusters (2006). This section expounds a critical comparative analysis between GFO-realism and *Smithian realism*.

Smith's position can be specified by the following conditions and assumptions, presented in Smith (2004, 2006), Smith and Ceusters (2006):

- Universals have an observer-independent objective existence; they are invariants of reality.
- Bad ontologies are those whose general terms lack a relation to corresponding universals in reality, and thereby also to corresponding instances.
- Good ontologies are representations of reality. A good ontology must be based on universals instead of concepts.

Unfortunately, there is a gap and a fundamental obscurity pertaining to condition 3. No definition for *reality representation* is provided. This fundamental

⁹If the theory is sufficiently expressive then an absolute consistency proof, based on finitary methods, is impossible. Hence, consistency proofs have a relative character; they are based on the method of formally reducing the considered theory to another already established theory whose consistency has a higher degree of evidence.

gap can never be closed without the use of concepts, i.e. there is no representation of reality without concepts.

To close this gap, let us construct an ontology *Ont* that may represent a part *D* of reality. Assume that there are universals $U(1), \dots, U(k)$ in *D* having an observer-independent objective existence; they are related, by assumption, to certain invariants in *D*, say $Inv(1), \dots, Inv(k)$, of reality. Consider, as an example, the universal *Ape*, denoted by *A*. There are individual instances $a(1), \dots, a(m)$ of *A* being individual apes. The universal *A* may be analysed and discussed, and information about *A* may be communicated. Hence, this universal is used and included in natural language and must be represented by a term, denoted by a word $w(A)$. $w(A)$ is a string in a language (spoken or written string), but it is more than a senseless string, because it denotes something. If a person uses $w(A)$, then this word has a meaning, and understanding a word implies to have access to its meaning. Let $m(w)$ be the meaning of the word.

To clarify, the entities $w(A)$, *A* and $m(w(A))$ exist and are pair-wise distinct. Obviously, $w(A)$ exists, and *A* exists by assumption. The existence of $m(w(A))$ is demonstrated by the process of understanding. The perception of $w(A)$ ¹⁰ is immediately connected to a meaning, and this meaning must be an entity of the mind. The meaning of $w(A)$ allows for understanding the spoken announcement “*Yesterday an ape escaped from the Zoo*”, independent from the actuality of this event. Since the universal *A* is separate and outside the mind, and the meaning $m(w(A))$ exists within the mind, it may be concluded that *A* and $m(w(A))$ are distinct. It is also clear that $w(A)$ and $m(w(A))$ are distinct, because $w(A)$ is merely a string, and one could, through a thought experiment, replace $w(A)$ with any other string *S* by stipulating that *S* is connected to $m(w(A))$; this connection can be established by a learning process. Then, the string *S* plays the role of a designation as well as $w(A)$. Now, we must clarify a further phenomenon: We must distinguish different (individual) meanings depending on different recipients; this can be illustrated and explained by a relation *intension*($p, m(w)$) that is interpreted as: the person *p* grasps the meaning $m(w(A))$ in the course of perceiving the word $w(A)$. Hence, the meanings are dependent on the individual subjects, and we use the expression $m(p, w(A))$ to denote the meaning of the word $w(A)$ with respect to the person *p*. Hence, distinct recipients of a word possess distinct meanings. On the other hand, we assume that any speaker of this language who perceives the word $w(A)$ relates it to the same meaning. But these meanings must be different entities and we must clarify how the equivalence between different individual meanings is determined and how it is to be understood. In achieving this equivalence we need a further step and a new construct, the *common intentionality* which is the result of communication and society (Searle, 1995). The common intentionality constitutes an agreement between the subjects about the equivalence of the different individual meanings $m(p, w(A))$. This agreement forms the basis for a concept $conc(A)$, which is an abstract, atemporal entity. In

¹⁰One must distinguish between symbols and tokens. Only tokens, being physical instances of symbols, can be perceived and transmitted through space and time.

general, common intentionality establishes an equivalence relation between different individual meanings. Concepts are abstract entities which are associated to these equivalence classes, and conversely, individual meanings can be understood as mental representations of concepts.¹¹ The concept $\text{conc}(A)$ is based on the one hand on individual meanings, and on the other hand on common intentionality which is related to the social stratum. Hence, a concept participates in the mental as well as in the social stratum of the world. Obviously, $\text{conc}(A)$ is distinct from $\text{m}(p,w(A))$, $w(A)$, and A .

In sum, the nodes in an ontology are labeled by terms that denote concepts. Some of these concepts, notably natural concepts, are related to invariants of material reality. Concepts are represented in individual minds and are founded in society. The same is true for individuals to which individual concepts correspond. The interrelations between universals, concepts, symbols and society are realized by various relations, including the relation of correspondence (between concepts and universals, and individual concepts and real individuals), the relation of representation (between concept and individual mind), the relation of foundedness (between concept and society), and the instantiation relation. We summarize that the restricted view of Smithian realism cannot be an ontological-philosophical foundation for the field of conceptual modeling and, in particular, for computer-science ontologies.

14.2.6 Levels of Reality

We assume that the world is organized into strata, and that these strata are classified and separated into layers. We use the term *level* to denote both strata and layers. GFO distinguishes, according to Poli (2001), at least three ontological strata of the world: the material, the mental-psychological, and the social stratum. Every entity of the world participates in certain strata and levels. We take the position that the levels are characterized by integrated systems of categories. Hence, a level is specified by a meta-level category whose instances are categories of certain kinds. Every level includes individuals too; these belong to the individual base of those categories specifying the level.

Among these levels specific forms of categorial and existential dependencies hold. For example, a mental entity requires an animate material object as its existential bearer. These levels are needed to describe adequately the entities of the world. Another problem is how the levels within of a single entity are organized, and how these levels are connected and how they are separated. It turns out that the separation cannot be precisely defined, because the phenomenal world that can be immediately perceived is already cognitively influenced. We assume that there is a physical level, denoted by PhysW , that is completely independent from the subject. Then there is

¹¹The study of mental representations of concepts is an important topic of cognitive psychology and cognitive linguistics. The theory of prototypes is an influential approach in this area of research (Rosch, 1975).

the phenomenal world, denoted by PhenW, which can be immediately experienced by perception.

Perception can be understood as realizing an immediate connection between the subject and the material, subject-independent, objective world. Natural concepts, based on perception, are the most primitive ones and additional concepts are constructed from them. The construction of more complex systems of subject-object interrelation,¹² as, for example theories, increase the distance between the subject and the perceived material world; on the other hand, they provide a deeper understanding of the world. One may think of several ontological layers which connect the subject with the independent reality. We consider the layer of perception as the mediator between the subject and reality and stipulate that the phenomenal world belongs to the material level.

14.3 Meta-Ontological Architecture of GFO

GFO has a three-layered meta-ontological architecture comprised of (1) a basic level consisting of all relevant GFO-categories and relations, (2) a meta-level, called abstract core level containing meta-categories over the basic level, for example the meta-category of all categories, and finally (3) an abstract top-level including *set* and *item* as the only meta-meta-categories.

The notion of a meta-category is a generalization of the notion of a meta-set or a meta-class in the set-theoretical sense. Let X be a set of entities, then every category C having exactly the entities of X as its instances is called a *categorial abstraction* of X. Usually, there can be several distinct categorial abstractions over the same set of entities. If a set X of entities is specified by a condition C(x), then the expression C(x) expresses a category which can be understood as a categorial abstraction of X.

A *categorial similarity abstraction* of X specifies properties that are common to all members of the set X. The specification of such categorial similarity abstractions in a language uses conjunctions of atomic sentences representing – in many cases – perceivable properties. There are also disjunctive conditions, for example the condition *x is an ape or x is a bridge*; obviously, the set of instances of this condition cannot be captured by a similarity abstraction. More complicated are categorial abstractions over categories, for example the category *species* in the field of biology.

The *abstract top ontology* (ATO) of GFO contains two meta-categories: *set* and *item*. Above the abstract top level there is a non-formal level, which might be called

¹²There is the general problem where the cut is made and defined between the subject and the independent real world. Several options are possible. Our approach can be justified by interpreting the phenomena as realization of dispositions of the objective independent world. These dispositions need a subject to come to appearance, more precisely, these appearances are realizations of dispositions within a subject. Hence, the phenomenal world is, on the one hand, anchored and founded in the objective reality, on the other hand it is realized in the subjective world. This connection is the basis for GFO's *integrative realism*.

philosophical level. On this level, several distinct, philosophically basic assumptions are presented, mainly around the notions of existence and dependency. The abstract top level is used to express and model the lower levels of GFO by set-theoretical expressions. To the abstract top level two basic relations are associated: membership \in and identity $=$. The abstract top level of GFO is represented by a weak fragment of set theory, and some weak axioms connecting sets with items. Among the axioms concerning sets belong the following:

$$\begin{aligned} \forall x y (Set(x) \wedge Set(y) \rightarrow (x = y \leftrightarrow \forall u (u \in x \leftrightarrow x \in y))) \\ \forall x y (Set(x) \wedge Set(y) \rightarrow \exists z (Set(z) \wedge z = \{x, y\})) \\ \forall x y (Item(x) \wedge Item(y) \rightarrow \exists z (z = \{x, y\} \wedge Set(z))) \end{aligned}$$

The *abstract core level* of GFO, called *abstract core ontology* of GFO (abbreviated ACO), exhibits the upper part of GFO. The abstract core ontology of GFO must first be determined by their main entity types and the relations among them, for which a certain vocabulary must be introduced. The entities of the world – being represented on the ATO-level by the items – are exhaustively divided into *categories* and *individuals*, and individuals *instantiate* categories. Moreover, among individuals we distinguish *objects*, and *attributives*.

By introducing a vocabulary for the considered entities we obtain the following signature: $Cat(x)$ denotes the meta-category of all categories, $OCat(x)$ represents the category of all object categories, $Prop(x)$ indicates the category of all properties, and $Rel(x)$ identifies the category of all relations. $Ind(x)$ is the category of all individuals, $Obj(x)$ designates the category of all objects, $Attr(x)$ represents the category of all attributives, $Rol(x)$ identifies the category of all roles, and $Rel(x)$ denotes the category of all relators. These categories are all presented as predicates, i.e. they occur on the ATO-level as sets of items.

The *basic level ontology* of GFO contains all relevant top-level distinctions and categories. All basic relations and categories are presented as set-theoretical relations and set-theoretical predicates. Categories which are not contained within the basic level we call domain categories. Domain categories are related to a certain domain D of the world, and on the domain level they are not presented and considered as sets, but as entities of its own. Formally, the vocabulary at the basic level of GFO is extended by additional constants denoting proper categories or individuals. If, for example, C denotes a domain category we write $x::C$ instead of $C(x)$, indicating that x is an instance of C . Domain categories may be linked in a simple way to the basic level predicates of GFO, using domain-upper linking axioms.

14.4 The Basic Categories of Individuals of GFO

In this section the basic classification of individuals is expounded. Full GFO is intended to include also an ontology of categories; this topic is outside the scope of the current paper.

14.4.1 Space-Time

The GFO approach of time is inspired by Brentano's ideas (1976) on continuum, space and time. Following this approach, chronoids are not defined as sets of points, but as entities *sui generis*.¹³ Every chronoid has exactly two extremal and infinitely many inner *time boundaries* which are equivalently called *time-points*. Time boundaries depend on chronoids (i.e. they have no independent existence) and can *coincide*. Starting with chronoids, we introduce the notion of *time region* as the mereological sum of chronoids, i.e. time regions consist of non-connected intervals of time. Time entities, i.e. time-regions and time-points, share certain formal relations, in particular the part-of relation between chronoids and between time-regions, denoted by $\text{tpart}(x,y)$ the relation of being an extremal time-boundary of a chronoid, denoted by the relations $\text{lb}(x,y)$ (x is left-boundary of y), $\text{rb}(x,y)$ (x is right boundary of y), and the relation of coincidence between two time-boundaries, denoted by $\text{tcoinc}(x,y)$.

Dealing with the coincidence of time boundaries is especially useful if two processes are to be modeled as "meeting" (in the sense of Allen's relation "meets"). In GFO there are at least three conditions that a correct model must fulfill:

- there are two processes following one another immediately, i.e. without any gaps,
- there is a point in time where the first process ends, and
- there is a point in time where the second process begins.

If, as is common practice, intervals of real numbers are used for modeling time intervals (with real numbers as time points), then these conditions cannot be simultaneously satisfied. In contrast, the Brentano approach allows for two chronoids to follow immediately, one after another, *and* to have proper starting- and ending-"points" by allowing their boundaries to coincide. The coincidence relation entails that there is no time difference between the coinciding time boundaries, while maintaining their status as two different entities. This way, conditions (a), (b) and (c) are fulfilled.

Many temporal concepts are based on a measurement function μ for chronoids. Then value $\mu(C)$ is called the duration of the chronoid C . Using μ we classify chronoids with respect to their duration, we may say that this chronoid has a certain duration α . Using a measurement function we may introduce a number of temporal concepts, for example days, minutes, and years. Analogously to chronoids and time boundaries, the GFO theory of space introduces *topoids* with *spatial boundaries* that can coincide. *Space regions* are mereological sums of topoids.¹⁴ To describe the structure of space (or of regions, respectively) we employ the basic relations *spatial part-of*, *boundary-of*, as well as the *coincidence of boundaries*. Formally, we use $\text{spart}(x,y)$ x is a spatial part of y , $\text{bd}(x,y)$, if x is a boundary of y , and $\text{scoinc}(x,y)$

¹³The GFO approach to time is related to what P. Hayes calls the *glass continuum* (Hayes, 1995). Furthermore, we advance and refine the theory of Brentano (1976).

¹⁴Again, we use ideas of Brentano (1976) and Chisholm (1983) for our theory.

if two (spatial) boundaries x and y coincide. This approach may be called *Brentano space*, and it is important to understand, that spatial boundaries can be found in a greater variety than point-like time-boundaries: Boundaries of regions are *surfaces*, boundaries of surfaces are *lines*, and boundaries of lines are *points*. As in the case of time-boundaries, spatial boundaries have no independent existence, i.e. they depend on the spatial entity of which they are boundaries.

14.4.2 Principal Distinctions

In this section we consider the most basic distinctions between individuals. Individuals are entities that are not instantiable, they are divided into space-time entities, concrete and abstract individuals. Concrete individuals exist in time or space whereas abstract individuals do not. Concrete individuals include this *cup*, or this *hundred meter run*, abstract individuals include the real number π or idealized prototypical individuals as, for example, canonical anatomical structures (Rosse and Mejno, 2003). With regard to the relationship between individuals and time and space, there is the well-known philosophical distinction between endurants and perdurants. An endurant is an individual that exists in time, but cannot be described as having temporal parts or phases; hence it is entirely present at every time-point of its existence and it persists through time. Perdurants, on the other hand, are extended in time and cannot be wholly present at a time-point. The definition of endurant and perdurant is based on Johnston and Forbes (1987), and Lewis (1986) where the notion of persistence is analysed and discussed.

According to this theory an entity persists if it exists at various times. The existence at various time can be understood – according to Johnston and Forbes (1987) – in two different ways. Something perdures if it persists by having different temporal parts at different times, though no one part of it is wholly present at more than one time; whereas it endures if it persists by being wholly present at more than one time. It turns out that the notion of endurant combines two contradicting aspects. If, for example, an endurant x is wholly present at two different time-points t and s , then there are two different entities “ x at t ” and “ x at s ”, denoted by $x(t)$ and $x(s)$, respectively. Now let us assume that x persists from $x(t)$ to $x(s)$. For example, newborn Caesar exists at time t , $x(t)$, while Caesar at age of 50 at s , $x(s)$. Then, persistence of implies that $x(t)$ and $x(s)$ are identical.

Unlike the vague notion of an endurant and perdurant we make a more precise distinction between *presential* and *process*. A presential is an individual which is entirely present at a time-point. The introduction of the term “presential” is motivated by the fact that presentials are individuals that may exist in the presence, where we assume that the presence has no temporal extension, hence, happens at a time-point. We introduce the relation $at(x,y)$ with the meaning the presential x exists at time-point y . In our approach we separate endurants into wholly present presentials and persisting persistents or perpetuants.

We pursue an approach which accounts for persistence using a suitable universal whose instances are presentials. Such universals are called *persistants*. These

do not change, and they can be used to explain how presentials that have different properties at different times can, nevertheless, be the same. They satisfy a number of conditions, among them the following: (a) every instance of a persistant is a presential; (b) for every time-boundary there is at most one instance which exists at this time-boundary; and (c) there is a chronoid c such that for every time-boundary of C the persistant has an instance at this time-boundary; and (d) every persistant is maximal, i.e. there is no proper categorial extension of it having the same extension. Further conditions should concern the relation of ontical connectedness and the relation of persistants to processes.

Persistants are special categories that can be instantiated. Are there individuals that correspond to persistants and take over some of its properties? We claim that for every persistant P of a certain subclass of persistants there exists an individual q called *perpetuant*, satisfying the conditions that it persists through time, and that it is related to the time-points of its duration by a relation $exhib(q,a,t)$. The relation $exhib(q,a,t)$ has the meaning that q exhibits the presential a at time-point t . A perpetuant is related to time by a set of time-points at which it exhibits presentials. A certain class of perpetuants, called material perpetuants, correlate to individuals which are sometimes called continuants. Unlike continuants – as a type of endurants – perpetuants are consistently presented.¹⁵ The existence of perpetuants are justified and supported by results of Gestalt theory Wertheimer (1912, 1922).

Processes have a temporal extension thus cannot be wholly present at a time-point. The relation between a process and time is determined by the projection function $ptime(p,c)$, having the meaning that the process p has the temporal extension C . C is called the framing chronoid of P . There is another basic relation for processes, denoted by e . The relation has the meaning that x is a process, t is a time-entity (a chronoid or a time-point), and the entity y results from the restriction of t to t . Two cases may be distinguished. If t is a chronoid, then y has the temporal extension t and is itself a process; y is a processual (or temporal) part of x . If t is a time-point, then y has no temporal extension, and, hence, cannot be a process. If e is wholly present at t then e is presential.

14.4.3 Material Structures

A material structure is an individual that satisfies the following conditions: it is a presential, it occupies space, it is a bearer of qualities, but it cannot be a quality of

¹⁵A perpetuant has - similar as a primitive universal - an implicit relation to time. The persistence of this kind of individual derives from its cognitive character. Persistence seems to be reasonable only for items that are invariant through a time-interval and at the same time are related at time-points of its duration to individuals which are immediately related to time and which may have different properties at different time-points. Such items are either special primitive universals or particular cognitive individuals. We do not apply the notion of persistence to abstract individuals, as to the number 100.

other entities, and it consists of an amount of substrate. Every material structure S occupies a certain space-region that exhibits the basic relation of S to space. The relation $occ(x,y)$ describes the condition that the material structure x occupies the space-region y . A material structure S is spatially contained in the space-region y , if the space-region x occupied by S , is a spatial part of y . In this case we say that x is the spatial location of S with respect to y or that y frames S . The relation $occ(x,y)$ depends on granularity; a material structure S , for example, may occupy the mereological sum of the space-regions occupied by its atoms or the convex closure of this system. We assume that in our considerations the granularity is fixed, and – based on this dimension – that the space-region occupied by a material structure is uniquely determined.

Using the relations $occ(x,y)$ and $spart(u,v)$ we define the relation $matpart(a,b)$ with the meaning that the material structure a is a material part of the material structure b .

$$matpart(x,y) \leftrightarrow \forall u \ v \ (occ(x,u) \wedge occ(y,v) \rightarrow spart(u,v))$$

Material structures may be classified with respect to the mereotopological properties of their occupied space regions. A material structure is said to be *connected* if its occupied region is a topoid. Every material structure consists of an *amount of substrate*. An amount of substrate may be a special persistent whose instances are distinct amounts at certain time-points; we call these *presential amounts of substrate*. An amount of substrate at a certain time-boundary, i.e. a presential amount of substrate, is always a part of the substrate of a material structure. The basic relation $consist(x,y)$ means the material structure x consists of the presential amount of substrate y .

Let x be a material structure which occupies a topoid T and let b be a spatial boundary of T . We postulate the existence of a material entity y which occupies the boundary b . These entities are called material boundaries, and they existentially depend on the material structure occupying the according topoid. Material boundaries are divided into *material surfaces*, *material lines* and *material points*. Every material surface is the boundary of a material structure, every material line is the boundary of a material surface, and every material point is a material boundary of a material line.

We introduce the basic relation $matbd(x,y)$ with the meaning x is a material boundary of the material structure y . Two material structures (or their material boundaries) touch if their occupied space regions have spatial boundaries with coincident parts. One has to take into consideration here that the spatial boundary which is occupied by a material boundary depends on granularity and context. Our notion of material structure is very general; almost every space-region may be understood as the location of some material structure. We single out material objects as material structures with *natural material boundaries*. This notion can be precisely defined. Let us consider a material structure S with material boundary B . A part $B(0)$ of B is a natural boundary if the two following conditions are satisfied:

- (1) There is a material structure T outside of S such that S and T touch at B(0).
- (2) A tangential part of S with boundary B(0) and a part of T touching S at B(0) can be distinguished by different properties.

Examples of such distinguishing properties are fluid, solid, and gaseous. As an example, let us consider a river. A river (at a time point of its existence, i.e. considered as a presential) is a material structure which consists of fluid substrate and has natural material boundaries at all places, with exception of the region of the river's mouth. The solid river bed may be distinguished from the river fluid and the river fluid may be distinguished from the air above the river. Within our framework certain puzzles can be easily solved. In Leonardo's notebooks there is mentioned: *What is it . . . that divides the atmosphere from the water? It is necessary that there should be a common boundary which is neither air nor water but is without substance, because a body interposed between two bodies prevents their contact, and this does not happen in water with air.* (cited in Casati and Varzi (1994)).

How can two things – the water and the air – be in contact and yet be separated? There are two material structures *W* and *A* (water and air), *W* consists of liquid substrate, *A* consists of gaseous substrate. *W* and *A* have natural boundaries because at the “touching area” we may distinguish *W* and *A* by the properties “fluid” and “gaseous”. These natural boundaries touch because their occupied space-boundaries coincide. The touching phenomenon is explained by the property described in the Brentano-space theory that pure space boundaries may coincide; they may be at the “same place” but, nevertheless, different. What is “interposed” between the two natural boundaries are two coinciding space-boundaries which do not occupy any space.¹⁶

14.4.4 Processual Complexes, Processes, and Occurrents

Processual complexes are the most general kind of concrete individuals which have a temporal extension. The temporal extension of a processual complex is a mereological sum of a non-empty set of chronoids. Processes form the most important sub-class of processual complexes, and occurrents centers around the notion of process. Occurrents are dependent entities that are related to processes in various ways. Some examples of processes or occurrents include: a rhinitis, seen as a sequence of different states of inflammation; writing a letter; sitting in front of a computer viewed as a state extended in time; the execution of a clinical trial; the treatment of a patient; the development of a cancer; a lecture in the sense of an actual event as well as a series of actual events, but opposed to the abstract notion of lecture; an examination.

¹⁶GFO presents a solution to a problem which arises in Smith and Varzi (2000). GFO gives a new interpretation of bona-fide boundaries in terms of natural boundaries. The claim stated in Smith and Varzi (2000) that bona fide boundaries cannot touch is counter-intuitive and ontologically false. A similar critics is sated in Ridder (2002).

14.4.4.1 Processual Complexes

A processual complex is a concrete individual whose temporal extension is a time region. The basic relation between temporal complexes and time is determined by the relation $ptime(tc, tr)$, where tr is the time-region which is associated to the processual complex pc ; we say that pc is projected onto the time-region tr . A processual complex is said to be connected if its projection to time is a chronoid. The relation $timerestr(x, t, y)$ has the meaning that the processual complex x , restricted to the time-structure t yields the entity y . The time-structure t is a temporal part of the time-region of x , and may additionally include an arbitrary set of time-points, selected from the projection of x . Then y is the temporal restriction of x to the time-structure t . Processual complexes are classified into coherent and non-coherent complexes.

14.4.4.2 Processes

The set of processes is a proper subset of the set of connected temporal complexes. Not every connected processual complex is a process because the latter satisfies a number of further conditions. The projection of a process p to time – described by the relation $ptime(p, c)$ – is a chronoid which is uniquely determined. Hence, the relation $ptime(p, c)$ establishes a function from the set of all processes to the set of all chronoids.

Just as parts of chronoids can be chronoids themselves, we assume that parts of processes are always processes themselves. If p is a processual part of the process q , denoted by $procpart(p, q)$, then the temporal extension of p is a temporal part of the temporal extension of q . Two processes p, q meet, denoted by $procmeet(p, q)$, if their corresponding chronoids temporally meet. If there is a process r such that p, q are processual parts of r , and the temporal projection of r is the mereological sum of the temporal projections of p and q , then r is said to be the processual sum of p and q . We stipulate that the processual sum of two processes – if it exists – is uniquely determined.

If a process P is restricted to a time-point of its temporal extension then the resulting entity cannot be a process, because it has no temporal extension. If this entity is a presential then it is called a boundary of the process. The relation $procbd(p, t, e)$ has the meaning that p is a process, t a time-point of the temporal extension of p , and e a presential at time-points t being the restriction of P to t . We assume that e is uniquely determined. Processes may be classified with respect to their boundaries. P is a quality process if any boundary of P presents an aggregate of qualities. Material processes contain in any of its boundaries a material structure.

Every process is *coherent* and *coherence* of a connected temporal complex implies that its boundaries are ontically and causally connected by the relations $ontic(x, y)$ and $caus(x, y)$. Coherence is a basic notion that cannot be defined and reduced to other concepts; it must be characterized by axioms, and these axioms are based upon our intuitions and experience of the phenomenal world. The

relation $ontic(x,y)$ is considered and stipulated as a primitive basic relation, hence, we assume that it cannot be defined by other relations. This relation can be illustrated and elucidated by examples. Assume, we consider vase V at a certain time-point t , and suppose that V breaks down at a later time-point t_1 into three parts V_1, V_2, V_3 . Then, these parts are ontically connected to V . One aspect behind the ontic-relation is a general ontological law of conservation of substrate and matter. Another example, demonstrating the ontic-relation, is related to the ship of Theseus S . Suppose, that after some time during which replacements of parts of S were carried out two ships S_1, S_2 came into being. Then only that ship is ontically connected to S whose parts originate from the parts of S .

Another facet of coherence is causality. We assume that coinciding boundaries of a process are causally related. In particular, in a process any of its boundaries is determined by its past. Hence, in a coherent process a boundary cannot be replaced arbitrarily by another presential. Analogously, not every extension of a process is coherent. But, we assume that every process has a processual extension (which is, hence, coherent), and is at the same time an extension of a process. Hence, we postulate that every process can be prolonged to the future and to the past.¹⁷

Processes satisfy an ontological inertial principle that can be formulated as follows. A state (which is a particular process) prolongs to the same type of state unless there is a cause to change it. In summary: ontical connectedness, causality, and the ontological principle of inertia are satisfied for processes. Coherence is a very important principle for processes, without coherence the world would disaggregate in many isolated individuals.

A material process is a process whose boundaries are material structures. A structural layer q of some process p is a “portion” of p that may be explained by the following example. Let be p a 100 m run with eight participants. Then the whole run is a process P , and every of its runners exhibits, as a process Q for itself, a layer of P . But even the process, associated to one of the runner’s part, say his right hand, forms a layer of P .

The layer of a process can be understood as a particular part of it, captured by the relation with the meaning, that x is a layer of the process y . Further part-of relations of processes may be derived from them. We may consider, for example, those parts of a process P which are processual parts of a layer of P . Two layers of a process are said to be separated if there are no interlacements between them. We claim that every process can be decomposed into separated layers. Furthermore, we believe that any number of processes with the same temporal extension can be embedded into a process containing them as layers.

¹⁷We assume an eternal view on processes. If we are speaking about the future or the past then these are relative notions that are related to an observer.

14.4.4.3 Occurrents

Occurrents are classified into *events*, *changes*, and *histories*. These entities depend on processes and are relatively defined with respect to universals. In contrast to a general understanding of “change” as an effect, a *change* – in the framework of GFO – refers to a pair of process boundaries. These pairs occur either at coinciding boundaries, like “instantaneous event” or “punctual”, or at boundaries situated at opposite ends of a process of arbitrary extension. The enrollment of a student is an example for the first kind of changes, called *discrete*. It comprises two coinciding process boundaries, one terminating the process of the matriculation, one beginning the process of studying.

An example of *continuous* change is illustrated by the decline in the course of a rhinitis. If two boundaries of this process coincide, one may not be able to assign to them a difference to the severity of inflammation, but if one considers boundaries that belong to an extended part of the inflammation process, there will be a difference. Both notions of continuous and discrete change are relative to contradictory conditions between which a transition takes place. Frequently, these contradictions refer to pairs of categories that cannot be instantiated by the same individual.

Locomotions are another representative of continuous change. Here, the contradictory conditions refer to some change of the distance of the moving entity to some entity or frame of reference. Changes are defined relatively with respect to a universal U whose instances are presentials.

Relying on those universals, we finally arrive at the following relations: Discrete changes are represented by $\text{dischange}(p, e_1, e_2, u_1, u_2, u)$,¹⁸ where e_1 and e_2 capture the pair of coincident process boundaries.¹⁹ This relation implies that p is a process, u_1 and u_2 are disjoint sub-universals of u , such that e_1 and e_2 instantiate u_1 and u_2 , respectively. Note, that this implies instantiation of both e_1 and e_2 of u , which prevents expressing artificial changes, e.g. a change of a weight of 20 kg to a color of red. The conditions described about *dischange* are necessary conditions a discrete change should satisfy. We may derive from *dischange* a relation dischange_1 defined as follows:

$$\text{dischange}_1(x, y, z, u) =_{\text{df}} \exists u_1 u_2 \text{dischange}(x, y, z, u_1, u_2)$$

Note, that if u has no proper disjoint subuniversals then discrete changes with respect to u cannot exist. Furthermore, if C is a change relative to the universal u and $\text{ext}(u) \subseteq \text{ext}(v)$ then C is a change for v , too.

¹⁸The representation of a change could additionally mention also two sub-processes $p(1)$, $p(2)$, where both processes meet, and $e(1)$ is the right boundary of $p(1)$, and $e(2)$ is the left boundary of $p(2)$.

¹⁹Recall that “coincident process boundaries” refers to the fact that the respective time-boundaries coincide. It does not mean that the presentials themselves should coincide.

Any *continuous process* has no discrete changes. For the purpose of formalizing continuous changes, we consider a subprocess q of the process p . If q is a continuous change of p with respect to the universals u , denoted by $\text{contchange}(p,q,u)$, then the following conditions are satisfied. q does not contain any discrete change with respect to subuniversals of u , but any two non-coinciding boundaries of q can be distinguished by subuniversals of u . The mentioned conditions are necessary conditions that should be satisfied by any continuous change. But they are, we believe, not sufficient to adequately capture the notion of a continuous change. Continuous processes, and continuous changes in particular, must take into consideration some further conditions which are related to a measurement system that includes an ordering between certain universals. A complete theory of continuous processes and changes must be elaborated yet. A refinement and generalization of continuous changes takes into consideration the idea of observable or measurable differences between non-coinciding boundaries of a process. It might happen that not only coinciding boundaries cannot be distinguished, but also boundaries of sufficient small temporal distance. For this purpose we may introduce a universal $\Delta(\lambda)$ of chronoids of minimal duration λ that is employed in order to embody the idea of observable differences during chronoids of length $\rho \geq \lambda$, while the change does not allow the observation of a difference between boundaries whose temporal distance is smaller than λ . The predicate is intended to formalize this approach. Changes can only be realized in terms of ontical connectedness and persistants/perpetuants, in order to know which entities must be compared with each other to detect a change.

Events are entities that exhibit a certain behavior relative to a process; every event is a right boundary of a process. In describing events we introduce a relation $\text{event}(p, e, u_1, u_2, u)$, where e is the right boundary of p , u_1, u_2 are different universals (with disjoint extension) of the “same” kind of instances, i.e. they are subsumed by a certain universal u . Furthermore, every boundary of p left from e , within a certain end-segment of p , is an instance of u_1 , but e itself is an instance of u_2 . We present the example of cell-division demonstrating an event. Let us assume the process p is called cell-division. This process starts with one cell, and ends with two cells. In the course of the process there is a continuous deformation of the cell, and at any time-point before the event we find one cell, i.e. to any boundary left from the event, the property of being one cell is verified. Hence the distinguishing properties to be considered are “to be one cell” (as a connected whole) or “to be two cells”. We may consider the same property for the left boundary of a process. The left boundary of a process p is a starting event of p , with respect to the universals u, u_1, u_2 , denoted by if every boundary right from e , within a certain initial segment of p is an instance of u_1 , but e is an instance of u . We consider an example of a process that has a starting event and a (final) event. Let B be a pool and let us consider the universals $u_1 =_{\text{df}} B \text{ is empty}$, $u_2 =_{\text{df}} B \text{ is completely filled}$ with water, u_3 the universal: $B \text{ is non-empty but not completely filled}$. Then, the process of filling the pool B with water has a starting event e_1 (the empty B), and a final event e_2 , B is completely filled.

Since every process is prolonged in the future there arises the question which types of coinciding boundaries may occur. Furthermore, not every universal is suitable to establish changes and events. We restrict in this section to the case that the boundaries of the process are material structures. Such a material structure p may undergo many changes during its existence. Which kinds of change for p are possible? We collect some types of changes without claiming that this classification is complete.

- P may change its qualities, say colour, weight, form, size; these are individuals that inhere in P , and are genuinely unary, i.e. it do not need any relation to other entities.
- P may change its relation to space, i.e. may move in space or may change its form, such that the relation $\text{occ}(x,y)$ is changed.
- P may loose spatial parts or may unify with other material structures.
- P may change its relation to other entities, in particular, P may change its role.

In all these changes the type of the changed entity should be preserved. A colour, for example, should not change into a weight, a form should not change into a colour. Furthermore, different changes may be interrelated to each other, for example the change of form and morphology changes the occupied space. Some of these interrelations are causally founded, for example the relation between the temperature and the size of an iron poke.

Histories in GFO are related to processes. A history is a pair $(p, (a_i)_{i < k})$ whereas p is a process, and $(a_i)_{i < k}$ are presentials at certain time-points $(t_i)_{i < k}$ of the temporal extension of p such that these presentials are constituent parts of the associated boundaries of p . k is either a natural number or equals ω . As an example we consider a patient p . p can be considered as a process $\text{Proc}(p)$, and let us assume that his temperature is measured every day four times and during on month. Then the measured values belong to presentials which are exhibited at the time-points of measurement.

14.4.4.4 Basic Classification of Processes

In this section we investigate the immanent structure of processes based upon the types of change occurring in it. Using the notions of discrete and continuous change, but also states, processes can be subdivided according to the nature of changes occurring within a process and according to their combinations. First, there are processes in which all (non-coinciding) internal boundaries determine sub-processes that exhibit continuous changes. These are continuous processes which are described, for example, in mechanics (Hermes, 1959).

States. A process p is a state with respect to the universal u , briefly a u -state, if every boundary of p instantiates u . p is said to be a strong u -state if, additionally, there are no disjoint sub-universals of u_1, u_2 of u and no boundaries e_1, e_2 which are separated by u_1, u_2 , i.e. $e_1 :: u_1$, and $e_2 :: u_2$. Every strong u -state is a u -state, but not conversely. If p is a strong u -state, then there exists an extensionally minimal

sub-universal v of u such that p is a strong v -state. This is not true for u -states. Furthermore, if the universal u does not contain any proper sub-universal then any u -state is a strong u -state. Strong u -states are already determined by certain minimal universals.

Continuous Processes. A process p is said to be continuous if p has no discrete changes and p is the mereological sum of continuous changes and states. If a process p is continuous then the partition into continuous changes and states is not necessarily uniquely determined. An example is a circular motion of a body.

Discrete and Discrete-Continuous Processes. But discrete changes may alternate with periods without changes (based on the same universals). Those parts of a process without changes may be called a *state*, which constitutes its own type of process. States, however, are a notion as relative as changes. A process is said to be discrete if it composed of states and discrete changes. Discrete-continuous processes are formed of discrete processes and continuous parts, hence such a process is e mereological sum of discrete and continuous processes. A process is said to be discreteless if it does not contain any discrete change. Continuous processes are always, by definition, discreteless. In summary, three standard types of processes can be identified: *continuous processes* based on intrinsic changes, *states*, and *discrete processes* made up of alternating sequences of extrinsic changes and states or continuous processes.

General Processes. We now consider the case that no restriction is proposed (fixed) for the distribution of universals over the boundaries of a process. Let I be a chronoid, and $Bd(I)$ the set of all boundaries of I . Furthermore, let be $\{u_1, \dots, u_k\}$ a set of universals whose instances are presentials, we assume that these universals are pair-wise extensional disjoint. Let f be a function $Bd(I) \rightarrow \{u_1, \dots, u_k\}$. Does there exists a process P such that P has a temporal extension I and for every time-boundary t of $Bd(I)$ holds that $e(t)$ is an instance of $f(t)$? The classification of general processes is an open problem.

Another dissection of the category of processes is geared toward the complexity of the process boundaries in their nature as presentials. Consider a person walking compared to a clinical trial. In the first case, the process of walking focuses on the person only (and its position in space), whereas the clinical trial is a process with numerous participants and an enormous degree of complexity and interlacement. It is clear that every process is embedded in reality, so the walking is not separated from the world and could be considered with more complexity.²⁰ However, processes often refer to specific aspects of their participants, so that dividing simple and complex processes appears to be useful. A process is called *simple* if its process boundaries are simple presentials or even mere qualities of presentials. In contrast to simple processes, *complex processes* involve more than a single presential at their boundaries. A finer classification of simple processes (according to the nature of its presentials) could be *quality-process* and *material-structure-processes*. Processes

²⁰The categories of situations and situoids as discussed in this paper are a first attempt to account for this in a systematic manner.

are not directly related to space, but such a relation can be derived from the process boundaries (which are presentials).²¹ With material-structure processes, each boundary comprises exactly one material structure $e(t)$, where t denotes the corresponding time-boundary. In this case, the convex frame f of the topoid occupied by $e(t)$ can be defined, denoted by $(e(t), f)$. In order to assign some topoid to the overall process we consider the convex closure of every frame f which is assigned to some $e(t)$ for any time-boundary t in the duration of the process.

For complex processes, which involve a system of material structures and qualities, both approaches can be combined. First, the inherence closure of all qualities in each process boundary is derived. Then one can determine the convex closure for each of the material structures found. The final step integrates all topoids determined in this way within a single convex closure, which is then assigned to the complex process as its spatial location.

An material object is an artefact if it was designed and produced by a subject, a human being. Similarly, we introduce the notion of a artefactual process which is designed by human beings. Among them there are executions of software programs, or the realization of a plan to achieve certain goals.

14.4.5 *Attributives*

Attributives are dependent entities, they always need a bearer. Attributives include, among others, properties, relators, roles, functions, and dispositions. Examples of qualities are particular weights, forms and colors. A sentence like “This rose is red.” refers to a particular object, a rose, and to a particular quality, red. Objects and attributives are connected by the basic relation of inherence. Atomic attributives have no parts or sub-components, they include qualities and roles. Atomic attributives are classified into context-free and contextual. Contextual atomic attributives are always parts of complex attributives. Qualities are context-free atomic attributives, roles are contextual atomic attributives being parts of relators. Examples of roles are available through terms like parent, child or neighbor. Here, parent and child would be considered as a pair of interdependent roles. Apparently, these examples easily remind one of relations like “is-child-of”. Indeed, a composition of interdependent roles is a *relator*, i.e. an entity that connects several other entities.

14.4.5.1 *Properties*

Things can have certain characteristics, features. To express them, natural and artificial languages make use of syntactic elements like adjectives/adverbs, or attributes/slots, respectively. Examples are: the severity of a rhinitis (a severe or minor); the shape of a nose (bulbous, pointy, flattened); the size of a filing cabinet;

²¹This resembles the idea of “indirect qualities” in Masolo et al. (2002).

the size of a clinical trial (the number of participating patients); the number of centers comprising mono- and multi-center trials; the age of a patient (which may affect the inclusion or the exclusion in a trial); the reputation of a university.

In the following, we present the GFO account on properties, which consists of two parts: First, the distinction between abstract *property universals* and their concrete instances, which are called *property individuals*.²² Second, both property universals and property and individuals must be distinguished from their respective values. At the abstract (universal) level, we distinguish between property universals and their *values*, which include the difference between phrases like “the size of a cabinet” and “a big cabinet”. The first phrase refers to a certain aspect of the cabinet. The second phrase refers to a value of this property of the cabinet, which reflects a relationship between the property universal, x , and the same property as exhibited by another entity, y .

Values of property universals usually appear in groups which are called *value structures* or *measurement systems*. Each of these structures corresponds to some property universal. More intuitively, one could say that the property may be measured with respect to some measurement system. For instance, sizes may be measured with the values “small”, “big”, or “very big”, which are the elements of one value structure. This structure and the particular values of the sizes of, e.g. a cabinet and a desk, respectively, allow for comparison of their sizes.

The notion of a value structure of a property is similar to a quality dimension in Gärdenfors (2000).²³ Further, value structures are related to quality spaces in Masolo et al. (2003).²⁴ Note, however, that various types of value structures can be found for the same property. Of course, one is tempted to include all these value structures within one comprehensive or “objective” structure. The latter would cover all values, such that any other structure appears as a selection of values of the objective structure. Instead of this, we currently consider it better to have distinct value structures (e.g. based on some measurement instrument), which may afterwards be aligned and composed into a broader structure, than to have a pre-defined “objective” structure. One reason for our approach is that the precise objective structure is unknown for most properties (choosing real numbers as isomorphic may often comprise too many values). In addition, all measurement instruments are restricted to a certain range of values, which can be measured using this instrument.

Within a value structure, several levels of generality may be distinguished, but, preliminarily, we understand value structures to be sets of values. Often it appears that a notion of distance can be defined, and that certain layers of value structures are isomorphic to some subset of real numbers, which allows for a mapping of values to pairs of a real number and a unit, as in the case of “10 kg”.

Coming to concrete entities, one can observe, that e.g. size (“the size of a filing cabinet”) can be a property of other entities apart from filing cabinets, as it

²²In earlier texts these were referred to as “properties” and “qualities”.

²³Note that the term “property value” here resembles Gärdenfors’ notion of “property”, our “property” his “quality dimension”

²⁴A quality space consists of all “quales” (our property values) of some “quality” (our property).

is a universal. Hence the question arises whether the size of the particular cabinet and the size of some other particular entity is literally the same entity. To answer this question, we introduce the distinction between property universals and property individuals (regarding these two categories, note the terminological and conceptual affinity with Masolo et al. (2003)).

In our example, we can differentiate between two entities: “the size” and “the size of that cabinet”. The size is a property universal (as introduced above). Because it is a universal, it is independent of the filing cabinet. But apart from the universal, we find the particular size of the particular cabinet, which exists only in the context of this cabinet and therefore existentially depends on it. We call individuals of this kind *property individuals*. To say that an individual entity has a property means that there is a quality individual which is an instance of the property universal and that this property individual *inheres* in its bearer. So the “size of that cabinet” is a property individual that inheres in the cabinet, while “size” is a property universal, of which the quality is an instance.

We introduce *values of property individuals*, which are analogous to values of property universals. For example, big and small may be the values of the size universal, whereas a particular big or small of some cabinet is the value of an individual quality, namely the size of that cabinet. Values of property individuals are individuals instantiating the corresponding property universals’ values. Moreover, the particular value x is linked to a property individual y by the relationship.

It should be stated explicitly that values of property universals are not considered as specialisations of property universals. Properties themselves can be classified and subdivided in various ways. One natural way to classify perceptible properties is assigning them based on the way in which they are perceived. This leads to visible properties (like lengths and color), smells, tastes (e.g. sweetness, bitterness) and so on.

However, there are also more formal classification principles for properties, for instance, according to the categories of the characterized entities. The following subcategories of properties are preliminarily distinguished with respect to the categories their bearers belong to. Note that for each category a different subrelation of has-quality may be introduced, in order to integrate relationships that are fairly established.

- Qualities of material structures, e.g. the color of a ball,
- Qualities of processes, e.g. the average speed of an object’s movement, running for half an hour, and
- Qualities of qualities, e.g. a color’s hue or brightness.

14.4.5.2 Relations and Roles

Roles are common in modeling, yet they have lingered in the background and only in recent years have they attracted focused interest (cf. Boella et al., 2005), although there are much earlier approaches dealing with roles as a central notion, as in Bachman and Daya (1977). Initially, the term role calls to mind terms like student,

patient, or customer – all refer to roles. In a comprehensive analysis, roles have been investigated for integration into GFO (Loebe, 2003, 2005). Here we provide a compact introduction to the general understanding of roles as well as the current state of role classification.

General Approach

Starting with a *role* r , there are two directly related notions, namely *player* and *context*.²⁵ Each role q requires a player p and a context c . More precisely, r is one-sidedly existentially dependent on p , and mutually existentially dependent with c . Two basic relations connect entities of these types: *plays*, denoted as $\text{plays}(x,y)$, connecting a player x with a role y ,²⁶ and *role-of* ($\text{role-of}(x,y)$), which ties a role x to its context y . In terms of the “standard” role example of student, John plays the role of the student in the context of his relationship to his university. Other examples refer to John as an employee in the context of some company, or as a mover of some pen, in the context of that movement.

Moreover, apart from roles, players, and contexts, roles are often contrasted with *natural universals*,²⁷ cf. (Guarino 1992). While “student” is a role, “human” is not a role, but a natural universal that provides players for roles. Intuitively, roles can be distinguished from natural universals by their dependence on a context, whereas for natural universals, the context of the considered role is irrelevant.

Each of these categories discussed thus far are self-contained, in the sense that they do not provide insights on how they are related to other GFO categories in this work. To establish these links, we first note that there are individuals as well as categories of roles (and all other notions). For more specific relations, different types of roles need to be distinguished. This classification is based on the contexts of roles, because the coupling of roles and contexts is more tight than between players and roles, cf. Loebe (2005).

Based on the literature, the following categories serve as contexts in various role approaches: relations, processes, and (social) objects. Accordingly, we distinguish three role types with the following informal definitions:

- A *relational role* corresponds to the way in which an argument participates in some relation;
- A *processual role* corresponds to the manner in which a single participant behaves in some process;
- A *social role* corresponds to the involvement of a social object within some society.

²⁵Note that “context” here is just an auxiliary notion for introducing roles, instead of being presented in a profound ontological analysis.

²⁶The literature provides *fills* and *hasRole* as other common terms for the plays relation.

²⁷Other terms in the literature are *natural type* (Guarino, 1992), *natural kind* (Wilkerson, 1995), *phenomenon* (Sowa, 1984), *base classifier* in UML (Rumbaugh et al., 1999), and *basic concept* in Sunagawa (2005).

Here, we focus on the relationships to the general role notions identified above. Moreover, the given classification is not meant to be complete, i.e. other categories may be contexts, thus yielding further role types.

Relators are the contexts of relational roles, i.e. a relator can be decomposed into at least two relational roles which complement each other. Intuitively, the role-of relation seems like a part-of relation in this case. Because relational roles refer to exactly one player, the plays relation corresponds to has-property. Accordingly, relational roles are subsumed by the category of properties. Consider that the number two is a factor of four. This refers to a relator with two role individuals, one instantiating the role universal “factor”, the other instantiating “multiple”. The first of these role individuals is played by two, while four plays the second role individual.

The generality of relations regarding the entities they connect is reflected in the fact that players of relational roles cannot be restricted by any specific category; hence, the natural universal for relational roles in general is the category “entity”.

Processual roles have processes as their contexts. As such they are processes themselves, and one may identify them as special layers of a process, because role-of is understood as a part-of relationship (as in the case of relational roles). The plays relation is different from plays for relational roles, because here plays corresponds to participation in a process.

When John moves a pen, for example, the movement is a process in which John and the pen are involved, in different ways. Accordingly, the process can be broken into two roles, “the mover” and “the moved”. John plays the first role, the pen the second. Imagining John as a mime who pretends to move a pen should provide a natural illustration of the notion of processual roles.

The case of the mime further exemplifies an uncommon case of roles: a single processual role may itself form a context. Almost all role notions are relational in nature, in the sense that their contexts are composed of several roles. In contrast, processes that comprise only a single participant are understood as a processual role, and likewise, as a context. Considering the plays relation, the potential players of processual roles are restricted to persistants, because a persisting entity is required to carve out roles from processes.

Note that the similarities of relational and processual roles leads to a category of *abstract roles*. The latter is functionally defined as providing “a mechanism of viewing some entity — namely the player — in a defined context” (Loebe, 2005). Given this abstraction, we can now introduce a final type.

Social roles differ from abstract roles in that their understanding depends much less on their context. Instead, social roles come with their own properties and behavior, which is a common requirement in many role approaches in computer science, cf. (Steimann, 2000). For example, if John is a student, he is issued a registration number and gains new rights and responsibilities. From a philosophical perspective, this view is further inspired by Searle (1995) and the ontological levels of Poli (2001), see Section 4. Social roles are considered to be social structures in GFO, which is an analogous category to material structures, but in the social stratum. However, social roles also need a foundation on the material level, which in general role terms corresponds to the plays relation. For instance, the human John plays a

social role that is characterized by specific rights and responsibilities. Note that so far we do not exclude that social roles themselves may play other social roles; hence, there may be chains of the plays relationship that must ultimately terminate by a role played by a material structure.

The contexts of social roles are also social structures, which may be called societies or institutions, cf. (Searle, 1995). Accordingly, a rough similarity between role-of and part-of is present for social roles as well. However, there are complex interrelations among entities of the social stratum, and the ontology of this stratum requires much more work.

Given that the general approach to roles is initially independent of other GFO categories, as well as the diversity of individuals introduced as roles, leads us to question why all roles should fall within the same category. Stated differently, what should the intrinsic commonalities between processual and relational roles be? We must admit that there are none – a fact that lies in the nature of category “role” itself, because, under a meta-level perspective, all general role characteristics apply to “role” itself.

These meta-level aspects further relate to the account of roles given by Guarino (and colleagues), who characterizes “role” as a meta-category of relationally dependent and anti-rigid categories (Guarino and Welty, 2001; Masolo et al., 2004). The latter means that for each instance of a role category, it is not essential to instantiate that category. These criteria can be reconstructed in GFO, where relational dependence corresponds to our contexts and anti-rigidity must be re-interpreted in terms of player universals. Roles in GFO differ from this approach in the sense that (1) there are role individuals, and (2) it may be essential to play a role. For instance, it is essential that the natural number two is a factor of four, and it is likewise essential that each human is a child. Anti-rigidity thus does not hold for *every* player universal. Nevertheless, in most cases it is a useful indicator for detecting player universals, and thus roles.

14.4.5.3 Functions

We understand a function to be an intentional entity, defined in purely teleological terms by the specification of a goal, requirements and a functional item. Functions are commonly ascribed by means of the has-function relation to entities that, in some context, are the realizations of the goal, execute such realizations or are intended by a reliable agent to do so. Functions are considered to be intentional entities and, hence, they are not objective entities of the world, but agent-dependent entities that primarily belong to the mental and social strata.

Structure of Functions

The pattern of the specification of a function F , called a function structure, is defined as a quadruple, $Label(F)$, $Req(F)$, $Goal(F)$, $Fitem(F)$, where:

- $Label(F)$ denotes a set of labels of function F ;
- $Req(F)$ denotes the requirements of function F ;

- $Goal(F)$ denotes a goal of F ;
- $Fitem(F)$ denotes a functional item of F .

Except for the label, these are called the function determinants, and they determine a function. Labels are natural language expressions naming the function. Most commonly, they are phrases in the form “to do something”, e.g. “to transport goods”. The requirements of the function set forth all the necessary preconditions that must be met whenever the function will be realized. For example, in the case of the function “to transport goods from A to B”, goods must be present at location A. Functions are goal oriented entities – specifying a function requires providing the goal it serves. However, goals are not identified with functions, as in Chandrasekaran and Josephson (1997). The goal of the function is an arbitrary entity of GFO — referred to also as a chunk of the reality — that is intended to be achieved by each realization of the function. In the case of transporting goods, the location of the goods at B is the goal. The goal specifies only the part of the world directly affected (or intended to be affected) by the function realization. In our case, it is the relator of goods being located at B. Often a goal is embedded in a wider context, being a complex whole, e.g. a fact, configuration, or situation, called final state. A final state of a function includes the goal plus an environment of the goal, therefore making the goal more comprehensible. Here, it is the relator together with its relata, i.e. goods located in B.

Functions are dependent entities, in the sense that a function is always the function of some other entity, executing it. The functional item of the function F indicates the role of entities executing a realization of F , such that all restrictions on realizations imposed by the functional item are also stipulated by some goal of F . In the case of “to transport goods”, the functional item would be the role universal “goods transporter”. Entities are often evaluated against functions. This is reflected in GFO by the relations of realization and realizer. Intuitively, an individual realization of a function F is an individual entity, in which (and by means of which) the goal of F is achieved in circumstances satisfying the requirements of F . Take the example of function F “to transport goods G from Leipzig to Berlin”, and the individual process of transportation of goods G by plane from Leipzig to Berlin. In brief, we can say that the process starts when the requirements of F are satisfied, and ends by achieving the goal of F , which, therefore, is the realization of function F .

Realization of Functions

It is important to understand the difference between a function and a realization, in particular with regard to their specification. To specify a function and its structure one must state what will be achieved; representing a realization usually means specifying how something is achieved. Note that not all functions must be realized by a process, as in the above example. In fact, in GFO we do not interpret functions in terms of processes or behaviors as described in Sasajima (1995). Apart from functions that are typically realized by processes or behaviors, we also consider functions realized by presentials. Consider, for instance, a pepper moth with a dark

covering sitting on a dark bark. This situation is the realization of the function of camouflaging a moth.

In every realization we find entities that execute this realization. They may be identified by references to functional items. For example, for the function “to transport oxygen”, the role “oxygen transporter” is the functional item. Now consider an individual transport process, i.e. a realization, involving a single red blood cell. That cell has the role “oxygen transporter” within this realization. This fact gives rise to a new entity that mediates between the realization and the cell itself, namely the cell as an “oxygen transporter” (cell-qua-oxygen transporter). Such an entity is called the realizer of the function and is considered to be a qua-individual, i.e. an instance of a role universal.

Ascription of Functions

Functions are often ascribed to entities, e.g. the function of oxygen transport is assigned to a process of blood circulation. We assign functions to entities by the has-function relation, whose second argument is a function, and the first is one of the following:

- an entity that is a realization of the function, e.g. for the function of transporting oxygen, the process of blood circulation;
- an entity that plays the role of the realizer in a realization of a function, e.g. the red blood cell in the process of blood circulation;
- an entity intended to be a realization or a realizer of a function.

The third case especially refers to artifacts that often inherit their functions from the designer, who intends for them to realize particular functions. The function ascription of that kind is called intended-has-functions. Note that artifacts are not only understood to be entities playing the role of realizers, as, e.g. a hammer that plays a realizer of the function “to hammer nails”. Additionally, artifacts may play the role of realizations, e.g. the process of transporting goods, which is a realization of the transport function, may be an artifact as well. This holds true especially with regard to services.

The intended-has-functions have a normative character, which allows for assigning such functions to entities that possess them as malfunctions. In short, the entity that has an intended function *F*, but is neither a realization nor a realizer of *F*, is said to be malfunctioning. The flavors and more detailed specification of malfunctions and of other notions outlined above can be found in Burek et al. (2006).

14.4.6 Facts, Propositions, and Situations

With relations, relators and roles, all components of facts are available, such that a more formal approach can be established. Since relations are entities connecting others, it is useful to consider collections of entities and their relators. The simplest

combinations of relators and relata are *facts*. Facts are considered as parts of the world, as entities *sui generis*, for example “John’s being an instance of the universal Human” or “the book B’s localization next to the book C” refer to facts. Note that the existence of facts is not uncontroversial in the philosophical literature. Approaches span from the denial of facts on the one hand, to their acknowledgement as the most primitive kind of entity on the other, cf. (Armstrong, 1997; Wittgenstein, 1922).

Further, facts are frequently discussed in connection with other abstract notions like propositions (cf. Loux, 1998), which are not covered in depth here. However, what can be said about propositions is that they make claims about the existence or non-existence of facts. Therefore, truth-values are assigned to propositions and they can be logically combined. In contrast, facts do not have a truth value.

There are additional notions that are frequently mentioned in connection with facts, for example *states of affairs*, which have yet to be included properly in GFO. With respect to representations of facts and propositions, we intend to study and integrate results from *situation theory* as initiated by Barwise and Perry (1983). This study will consider notions like infons and situation types, and will comprise the integration of these notions with those mentioned herein, like propositions and facts.

Another aspect to be stressed refers to the kinds of entities which facts are about, as these are not necessarily individuals. For example, the fact “Mary is speaking about humanity” refers to a relator of type “speaking”, which connects Mary with the universal humanity. On the basis of the relator and the types of the arguments, several kinds of facts can be distinguished. Here, one immediate option is to look at the appearance of individuals (e.g. none, at least one, all) and categories. Facts that contain at least one individual are called *individual facts*, while non-individual facts are called *abstract*.

Individual and abstract facts may be further classified. We outline a refined classification that pertains to individual facts and is important for the category of situations and situoids. The basis of this classification is the temporal interrelationships of the individual constituents of facts. An individual fact is called a *presential fact* if all of its individual constituents are presentials, which exist at the same time-boundary. Facts that are not presential facts can still be classified in many different sub-types based on similar temporal criteria. Another dimension for classification is to refer to a finer classification of the constituents, like facts about presentials, facts about processes, mixtures of these, and so forth. The development of a practically relevant classification remains to be completed.

As yet, facts themselves have only been considered as individuals. However, it appears reasonable to speak of factual universals. For instance, sentences in the form “A man kisses a woman”, can be interpreted in a universal sense. Each relation R, gives rise to a factual universal F(R), whose instances are composed of a relator of R and its arguments. Altogether, every relator of R has a corresponding fact instantiating F(R). In this section we survey some basic notions about the most complex entities in reality, namely situations and situoids.

Material structures, properties, and relators presuppose one another, and constitute complex units or wholes. The simplest units of this kind are facts. A

configuration is an aggregate of facts. We restrict the discussion in this section to a special type of facts, and ask whether an aggregate of facts can be integrated into a whole. Put differently, we ask whether a collection of facts constitutes a whole. We consider a collection of presential facts which exist at the same time-boundary. Such collections may be considered to be presentials, and we call them *configurations*.

It is further required that configurations contain at least one material object. Material objects are entities having a natural boundary, and on this basis, configurations may be classified as either *simple* or *non-simple*. A simple configuration is a configuration that is composed of exactly one material object and has only properties inhering in that material object. A configuration is said to be non-simple if it is made up of more than one material object, and these are connected by relators.

A *situation* is a special configuration which can be comprehended as a whole and satisfies certain conditions of unity, which are imposed by relations and categories associated with the situation. We consider situations to be the most complex kind of presentials.

Configurations have a counterpart in the realm of processes, which we call *configuroids*. They are, in the simplest case, integrated wholes made up of material structure processes and property processes. Furthermore, there is a category of processes whose boundaries are situations, and that satisfy certain principles of coherence, comprehensibility and continuity. We call these entities *situoids*; they are regarded as the most complex integrated wholes of the world. As it turns out, each of the entities we have considered thus far, including processes, can be embedded in a situoid. A situoid is, intuitively, a part of the world that is a coherent and comprehensible whole and does not need other entities in order to exist. Every situoid has a temporal extent and is framed by a topoid. An example of a situoid is “John’s kissing of Mary”, conceived as a process of kissing in a certain environment which contains individuals of the persistants John and Mary.

Every situoid is framed by a chronoid and a topoid. We use here two relations $tframe(s,y)$, and $tframe(s,y)$. Note that the relation $tframe(s,x)$ is equivalent to $prt(s,x)$, since a situoid is a process. The relations $prs(s,x)$ and $sframe(s,x)$ are different.

Every temporal part of a situoid is a process aggregate. The temporal parts of a situoid s are determined by the full projection of s onto a part of the framing chronoid c of s . This full projection relation is denoted by $prt(a,c,b)$, where a is a situoid, c is a part of the framing chronoid of a , and b is the process that results from this projection. Boundaries (including inner boundaries) of situoids are projections to time-boundaries. We assume that projections of situoids to time-boundaries, which are denoted by $prt(a,t,b)$, are situations. In every situation, a material structure is contained, and we say that a presential e is a constituent of a situoid s , $cpart(e,s)$, iff there is a time-boundary t of s such that the projection of s onto t is a situation containing e .

Situoids can be extended in two ways. Let s, t be two situoids; we say that t is a *temporal extension* of s , if there is an initial segment c of the chronoid t such that the projection of t onto c equals s . We say that t is a *structural extension* of s if s is a structural layer of t (cf. Section 14.2). Both kinds of extensions can be combined to

form the more general notion of a *structural-temporal extension*. Reality can – in a sense – be understood as a web of situoids that are connected by structural-temporal extensions. The notion of an extension can be relativized to situations. Since there cannot be temporal extensions of situations, an extension t of the situation.

s is always a structural extension. As an example, consider a fixed single material structure p , which occurs in situation s . Every extension of s is determined by adding further qualities or relators to s to the intrinsic properties of p . A quality-bundle that is unified by the material structure p is called *saturated* if no extension of s adds new qualities. It is an open question whether there is an extension t of s , such that every material structure p in t unites with a saturated bundle of qualities.

A *configuroid* c in the situoid s is defined as the projection of a structural layer of s onto a chronoid, which is a part of the time-frame of s . In particular, every structural layer of s is itself a configuroid of s . Obviously every configuroid is a process. But not every process is a configuroid of a situoid, because not every process satisfies the substantiality condition.

We postulate as a basic axiom that every occurrent is – roughly speaking – a “portion” of a situoid, and we say that every occurrent is embedded in a situoid. Furthermore, we defend the position that processes should be analyzed and classified within the framework of situoids. Also, situoids may be used as ontological entities representing contexts. Developing a rigorous typology of processes within the framework of situoids is an important future project. Occurrents may be classified with respect to different dimensions, among them we mention the *temporal structure* and the *granularity* of an occurrent.

As a final note regarding situoids, configurations, and their relatives, there are a number of useful, derivable categories. For instance, one can now define situational histories as histories that have only situations as their boundaries. In general, the theory of these entities is considered a promising field for future research.

14.5 Basic Relations of GFO

In this section we summarize the basic relations of GFO.

14.5.1 Existential Dependency

Entity is the category of everything that exists. We consider the entity level as a philosophical level at which the most general distinctions are considered. These are distinctions of modes of existence and of existential dependency. For many types of entities, their instances existentially depend on other entities. For instance, a time-boundary depends on the chronoid it is a boundary of, or the quality that inheres in a material structure depends on that structure. Various types of dependency relations are discussed in the philosophical literature, see e.g. [Chapter 9](#) in Johansson (1989). It turns out that the notion of existential dependency is rather vague and needs

further investigation. The classical definition of *existential dependence* or *ontological dependence* is given by the following informal definition which is preliminarily adopted:

Definition: An entity x is ontologically dependent on y when x cannot exist unless the y exists.

14.5.2 Set and Set-Theoretical Relations

The membership relation is the basic relation of set theory. $\text{Set}(x)$ denotes the category of all sets, represented as a unary predicate. $x \in y$ implies that either x and y are both sets, or x is a so-called class-urelement and y is a set. The subset relationship \subseteq is defined in terms of membership. We include in the ontology of sets an axiomatic fragment of formal set theory, say of ZF, in particular, the axiom of extensionality: As sets can be nested, we can consider all set-urelements that occur in a set. First, there is the least flattened set $y = \text{trans}(x)$, which extends the nested set on the first level of nesting with all class-urelements contained in any depth of nesting. That means, y satisfies the conditions $x \subseteq y$, and for every $z \in y$ holds that $z \subseteq y$. Then the class $\text{supp}(x) = \{a \mid a \text{ is a class-urelement and } a \in \text{trans}(y)\}$, called the support of x , contains all class-urelements of x and only them. A class x is said to be pure if $\text{supp}(x) = \emptyset$. We defend the idea by Lewis (1986) that the ontological status of sets can be reduced to the singletons, i.e. to the understanding of the transformation providing the singleton $\{a\}$ from an entity a . In (Herre, 2010) some ideas about this topic are discussed.

14.5.3 Instantiation and Categories

$\text{Cat}(x)$ is a predicate that represents the (meta)-category of all categories. We do not consider Cat to be an instance of itself. The symbol $::$ denotes instantiation. Its second argument is always a category, the first argument can be (almost) any entity. If the second argument is a primitive category, then the first must be an individual. Individuals – in general – can be understood as urelements with respect to instantiation. Since we assume categories of arbitrary (finite) type, there can be arbitrarily long (finite) chains of iteration of the instantiation relation. Since sets have no instances (they have elements) they can be understood as another kind of urelements w.r.t. instantiation. On the other hand, categories do not have elements, but instances, hence categories are urelements with respect to the membership relation.

The definable extension relation, $\text{ext}(x,y)$, is a cross-categorical relation, because it connects categories with sets and is explicitly defined in the following way: $\text{ext}(x,y) = \text{Set}(y) \wedge \forall u (u \in y \leftrightarrow u :: x)$. We may stipulate the existence of the set of all instances of a category by the following axiom (existence axiom): $\forall x (\text{Cat}(x) \rightarrow \exists y (\text{ext}(x,y)))$. If we assume this axiom then we may define the extensionality operator for categories: $\text{Ext}(x) = \{y \mid y :: x\}$. Note, that the existence axiom

contradicts the foundation axiom for sets, in case of existence of non-wellfounded categories. For this reason, we do not assume the foundation axiom for sets.

14.5.4 Property Relations and Relators

Further, several relations connect properties (or individual property instances), their values and their bearers. If – for reasons of brevity – individual properties are called “qualities”, there are the general relations has-property $hprop(x,y)$, and has-quality, $hqual(x,y)$, which relate a property bearer x to one of its properties/qualities y . However, there are specializations for certain types of arguments. The best known of such specializations is the relation of inherence, $inh(x,y)$, to be a sub-relation of has-quality. The phrase “inherence in a subject” can be understood as the translation of the Latin expression “in subjecto esse”, as opposed to “de subjecto dici”, which may be translated as “predicated of a subject”. Sometimes inherence is called ontic predication. The second kind of relations connects a property with some value of a measurement system. In the denotation $value(x,y)$ x refers to the property/quality y to the value. Relators are instances of relations. The role-of relationship was introduced as a close relative of part-of. It relates roles x and their contexts y , denoted by $roleof(x,y)$. Thus far we have introduced role-of between processual roles and processes and between relational roles and relators.

14.5.5 Parthood Relation

Part-of is a basic relation between certain kinds of entities, and several relations have a similar character.

Abstract and Domain-specific Part-of Relations. The abstract part-of relation is denoted by $p(x,y)$, while the argument-types of this relation are not specified, i.e. we allow arbitrary entities to be arguments. We assume that $p(x,y)$ satisfies the condition of a partial ordering. Domain-specific part-of-relations are related to a particular domain D , which might be the set of instances of a category. We denote these relations as $part_D(x,y)$. We assume that for a domain D , the entities of D and its parts are determined. There is a large family of domain-specific part-of relations, the most general of these are related to basic categories as $Chron(x)$, $TReg(x)$, $Top(x)$, $SReg(x)$, $MatS(x)$, $Proc(x)$. In the following sections we provide an overview of the most important category-specific part-of relations.

Part-of-Relation for Sets. We hold that the part-of-relation of sets is defined by the set inclusion, hence $part-set(x,y) := Set(x) \wedge Set(y) \wedge x \subseteq y$. If we assume the power-set axiom for sets, then the mereology of sets corresponds to the theory of Boolean algebras.

Part-of-Relations for Time and Space The part-of relations of time and space are related to chronoids, time-regions, topoids, and space regions. We introduce the unary predicates $Chron(x)$, $TReg(x)$, $Top(x)$, $SReg(x)$, and the binary relations

$t\text{part}(x,y)$, $s\text{part}(x,y)$. Every notion of part-of allows for a non-reflexive version of the relationship, which expresses proper parthood. These are denoted by adding a “p” to the above predicates, e.g. $pp(x,y)$ or $tppart(x,y)$. In particular, $s\text{part}$ applies to spatial regions, $t\text{part}$ refers to time regions and chronoids, while $c\text{part}$ represents a relationship between situoids (or situations) and their constituents. The constituents of a situoid s include, among other entities, the pertinent material structures (that participate in s) and the qualities that inhere in them. Further, facts and configurations are constituents of situoids. Not every part of a constituent of a situoid, however, is contained in it.

Part-of-Relation for Material Structures The basic relations pertaining to material structures are $\text{MatS}(x)$, for “ x is a material structure”, and $\text{matpat}(x,y)$, which means that the material structure x is a part of the material structure y . We assume among the basic axioms:

$$\forall x y u v (\text{MatS}(x) \wedge \text{matpart}(x,y) \wedge \text{occ}(x,u) \wedge \text{occ}(y,v) \rightarrow \text{spart}(v,u))$$

We stipulate that the relation $\text{matpart}(x,y)$ is a partial ordering, but additional axioms depend strongly on the domain under consideration.

Part-of-Relation for Processes. The part-of relation between processes is denoted by $\text{procpart}(x,y)$, meaning that the process x is a processual part of the process y . We assume the basic axiom:

$$\forall x y (\text{Proc}(x) \wedge \text{procpart}(y,x) \wedge \text{prt}(x,u) \wedge \text{prt}(y,v) \rightarrow \text{tpart}(v,u))$$

$\text{prt}(x,u)$ states that the process x has the temporal extension u , or that the process x is temporally projected onto u . Again, we stipulate that the relation $\text{procpart}(x,y)$ is a partial ordering, but additional properties of this relation depend on a concrete domain. For example, in the processes of surgery, only certain processual parts are relevant.

14.5.6 Boundaries, Coincidence, and Adjacency

We do not consider boundaries as being parts of entities. The boundary-of relationship connects entities of various categories, namely (a) time-boundaries and chronoids, (b) spatial boundaries and space regions, (c) presentials and processes, and (d) material boundaries and material structures.

We have not introduced a general relationship, but particular boundary-relations for each of these cases. Case (a) relies on the notions of left and right boundary-of, $\text{lb}(x,y)$ and $\text{rb}(x,y)$, respectively. In case (b), $\text{bd}(x,y)$ denotes the fact that x is a spatial boundary of y . Case (c) is discussed in the section on time and space, whereas the fourth case is not yet formalized.

Coincidence is a relationship between space boundaries or time boundaries, respectively. Intuitively, two such boundaries are coincident if and only if they

occupy “the same” space, or point in time, but they are still different entities. Obviously, congruence of extended boundaries like surfaces is entailed by their coincidence.

Further, the notion of coincidence allows for the definition of *adjacency*. In the case of space-time-entities, these are adjacent as soon as there are coincident parts of their boundaries. In contrast, material structures and processes cannot have coincident boundaries. Nevertheless, they are adjacent if the projections of their boundaries are adjacent.

14.5.7 Relations of Concrete Individuals to Space and Time

Concrete individuals have a relation to time or space.

Material Structures. Material structures are presentials, hence they exist at a time-point, and the relation $at(m,t)$ captures this relation. The relation $at(m,t)$ is functional, hence a presential m cannot exist at two different time-points. The binary relation of *occupation*, $occ(x,y)$, describes a fundamental relation between material structures and space regions. Occupation is a functional relation because it relates an individual to the minimal topoid in which a material structure is located. *Location* is a less detailed notion, which can be derived in terms of occupation and spatial part-of. An x is located in a region y , $loc(x,y)$, iff the topoid z , occupied by x , is a spatial part of y . Every process has a temporal extension. This temporal extension is called the projection of the process to time, and is denoted by $prt(x,y)$. We distinguish several cases: $prt(x,c)$, $at(y,t)$, $prb(x,t,y)$, where x is a process, y is a presential, c is a chronoid, and t is a time-boundary. The binary relations assign a temporal entity to presentials and processes, while $prb(x,t,y)$ is the projection of a process x to its boundary y , which is determined by the time-boundary t . Note that prb can be used to define the relations $at(x,y)$ and $partic(x,y)$.

Every situoid, for example the fall of a book from a desk, occurs over time and occupies a certain space. The binary relations of *framing*, such as $tframe(s,c)$, $sframe(s,x)$ binds chronoids C or topoids x to situoids s . We presume that every situoid is framed by exactly one chronoid and one topoid. The relation $tframe(s,z)/sframe(s,z)$ is to be read: “the chronoid/topoid z frames the situoid s ”.

14.5.8 Participation

Participation relates individuals to processes. There are several forms of participation of an individual in a process. $particpres(x,p)$ means that the presential x participates in the process p . This is the case if the restriction of p to a certain time-point contains x . The relation $particperp(x,y)$ states that the perpetuant x participates in the process y . This is the case if every presential z which x exhibits stand in the

relation *participres(z,y)*. The participance of a persistant x in a process y is defined analogously. A process x participates in the process, denoted by *particproc(x,y)* if x is a layer-part of x .

14.5.9 Association

The relation *assoc(s,u)* means “the universal u is associated with the situoid s ”. These universals determine which material relations and individuals occur as constituents within a given situoid. Thus, the association provides information about the granularities and viewpoints that a situoid presupposes. For example, a situoid s may be a certain part of the world encompassing the life of a tree in a certain environment. If a tree is considered as an organism, then the universals associated with s determine the viewpoint of a biologist, and the associated granularity of included types of individuals (branches are included, electrons are not). The association relation is related to a cognitive procedure that transforms the mere material structures into situations and situoids. Situations and situoids are parts of the world that can be “comprehended as a whole”. At the purely material level, these parts can be understood – we believe – as superimposing fields (gravitational, electromagnetic, etc.), which constitute a certain distribution of energy and matter. At the mental or psychological level, this distribution is perceived as a material structure. A material structure – as we have introduced it – is a pre-version of a situation. At this level of perception, certain structures may already be perceived: material boundaries, colors and the like. The level of comprehension, of understanding this part of the world as a situation, needs more than only the elementary perceptual structures. Comprehension presupposes the availability of concepts, and the formation and the use of concepts seems to be a component of the mind’s cognitive process. The association relation is related to this ability of the mind to understand material structures of the world as situations.

14.5.10 Ontical Connectedness and Causality

Presentials are connected by spatio-temporal and causal relationships, which give rise to persistants and perpetuants. The relation *ontic(x,y)* connects x and y by an integrated system of such relationships. It is assumed that x and y are processes or presentials. We believe that there are different relations of this kind. One interesting case of ontical connectedness is substrate-connectedness. Two material structures x and y are substrate-connected if they consist of the same amount of substrate. For example, a statue s made of clay, considered at a certain time-boundary, is substrate-connected with the material structure that results from a crash, which destroys s . In the present stage of investigation of *causality*, the relation between *causes* and their *effects* is seen as a special relation between presentials (contrary to the DOLCE account as given in Lehmann et al., (2004)). This basic relation shall support the

traditional intuitions of regularity, counterfactual dependency and manipulability. In a second step, the basic causal relation is then extended to cover processes as causal relata as well.

14.6 Object-Process Integration

In this section we study the inter-relations between processes and other entities. In particular, we propose a framework for integrating several aspects of objects and processes into one system.

14.6.1 Processual Unification and Cognition

In GFO, spatio-temporal processes are independent individuals of reality; they exhibit the most fundamental of its categories.²⁸ All other categories of individuals are built upon them. Hence, processes unify the world of spatio-temporal individuals. Processes establish the coherence of the world; without processes the world would disintegrate into numerous separate and isolated entities. Presentials existentially depend on processes; we hold that every presential is a part of a boundary of a process. Hence, we stipulate the following axiom:

$$\forall x(Pres(x) \rightarrow \exists yz(proc(y) \wedge procbd(z, y) \wedge cpart(x, y))$$

A presential x participates in the process y , denoted by $particpres(x, y)$, if a boundary of y contains x as a constituent part. We postulate that for every time-boundary t of the temporal extension of a process p there exists a presential which is the boundary of p at t . From this follows the condition

$$\forall xy(Proc(x) \wedge procbd(y, x) \rightarrow Pres(y))$$

Let $Bd(p)$ be the set of boundaries of the process p . Every categorical abstraction over $Bd(p)$ is called a persistant of p . Hence, a persistant is a concept whose instances equals the set of boundaries of a process. A persistant U of a process p is called maximal if there is no persistant V of p containing U as a proper intensional sub-concept. We hold that for every process there exists a greatest persistant which is uniquely determined. Persistants present the phenomenon of persistence through time with respect to the boundaries of a process; they capture those features of the boundaries of a process which do not change through time.

²⁸Processes, as other individuals, are not completely free of cognition. To clarify this situation we introduced in Section 14.2.4 layers between the subject and reality. The layer of perception connects the subject with reality and we stipulate that the phenomenal world, though cognitively biased, belongs to the reality outside the subject.

A process is said to be material if every boundary of it is a material structure; the associated persistants are called material. Perpetuants are individuals that exhibit a cognitive construction built upon particular material persistants; the existence of such cognitive individuals is supported by results and methods of Gestalt theory (Wertheimer, 1912, 1922).²⁹ Perpetuants are related to those entities that are sometimes called continuants or objects, as apples, cars or houses. Unlike persistants, being concepts, perpetuants are individuals which have an indirect relation to space and time.

14.6.2 Completed Categories and Integrated Individuals

The spatio-temporal individuals of the world are classified into processes, presentials and perpetuants. Persistants are universals which present categorial abstractions of the set of boundaries of a process. The complete specification of a material structure with closed boundary, say, of an ordinary object, integrates four aspects into one system: the object as a presential, as a process, as a perpetuant, and as a persistant. We explain and demonstrate this interrelation and integration by an ontological analysis. Consider an everyday name like “John”. What does John refer to in an ontologically precise sense? There are, obviously, four possibilities, i.e. four entities of different categories:

- John denotes a presential $Pres(John,t)$ at some point t in time,
- John refers to a perpetuant $Perp(John)$,
- the name is given to a process $Proc(John)$,
- John refers to a persistant $Perst(John)$.

Starting with an act of perception of John, we assume that a presential is recognized at a time-point t , call it $Pres(John,t)$. If one has seen John several times, with probably varying properties, but still being able to identify him, this forms the basis for a perpetuant and a persistant, say $Perp(John)$, and $Perst(John)$. Now, one may consider the extension of this persistant (which is a universal), i.e. the set $Ext(Perst(John)) = \{J|J :: Perst(John)\}$, and analogously, the set $Exhib(Perp(John)) = \{J|exhib(Perp(John),t,J)\}$. Obviously, the entity $Pres(John,t)$ referred to above is a member of this class. Also, one can say that any two members of that class represent “the same John”.

In the fourth interpretation, the name John denotes a process $Proc(John)$ of a special kind. We postulate the existence of a process $Proc(John)$ whose set of restrictions to its time-boundaries equals the set of instances of $Perst(John)$ and the set $Exhib(Perp(John))$. Furthermore, we see that the presentials associated to John can be derived from a process by taking the restrictions of this process to

²⁹Persistants apply to every process, whereas the construction of perpetuants is restricted to a particular class of material processes.

time-boundaries. On the other hand, the persistent $\text{Perst}(\text{John})$ or the perpetuant $\text{Perp}(\text{John})$ cannot be directly derived from a process because a categorial abstraction and a cognitive construction must be taken into consideration. The entities $\text{Perp}(\text{John})$ and $\text{Perst}(\text{John})$ capture important aspects of John's personal identity.³⁰ We call $\text{Int}(\text{John}) = (\text{Proc}(\text{John}), \text{Perp}(\text{John}))$ an integrated individual.

A complete understanding and description of concrete individuals needs all four aspects specified in our integrative system. If one of these aspects is missing we will face problems. If, for example, we consider John as a persistent or perpetuant only, then this John cannot engage in any temporal action, for example, the activity of eating. John's actions and activities are realized on the process level. If we consider John as the set of all presentialist Johns, then we have the same problem; since any action takes time, but a presentialist John cannot carry out any action. If John is a process only, then the problem becomes identifying the boundaries of the process because any natural process may be prolonged both into the future and into the past. Furthermore, we perceive John as a presential, which is missing in a pure processual understanding. We face similar problems pertaining to a full understanding of concrete entities, if we combine only two of the above aspects.

The notion of an integrated individual can be generalized to categories. Let us consider, for example, the category E of elephants. We may associate to E three basic categories: $\text{Proc}(E)$, the category of all processes spanned by all individual elephants, $\text{Pres}(E)$ the class of all presentialist elephants, $\text{Perp}(E)$ the class of all perpetuant elephants. The completed category of elephants, denoted by $\text{Compl}(E)$, has the three instances $\text{Proc}(E)$, $\text{Perp}(E)$, $\text{Pres}(E)$; hence $\text{Compl}(E)$ is a higher order category. $\text{Compl}(E)$ can be extended by adding a class of persistants associated to the elephants. Hence, the system $(\text{Compl}(E), \text{Perst}(E))$ exhibits the complete informations about the category of elephants.

14.6.3 Comparison to Other 4D-Ontologies

GFO is basically a 4D-ontology, in the sense, that the processes form the most fundamental category of individuals. Furthermore, processes cannot be considered as mere aggregates or sets or mereological sums of their boundaries. Hence, our theory of boundaries differs from the theory of stages in a 4-dimensional setting in the spirit of Sider (2001). Further, GFO adds to the pure 4D-view a cognitive level at which perpetuants and persistants are introduced; these correspond to entities called continuants. Persistants can be introduced for every process p by a categorial abstraction over the set $\text{Bd}(p)$ of boundaries of p .

³⁰A full elaboration of our approach to personal identity is much more complicated. It must consider the underlying process, the place of consciousness and will, and the dynamic interrelations between the persistent, the perpetuant, the presentials, and the process.

14.7 Principles of Ontology Development and Ontological Modelling

The application of GFO as a framework for conceptual modelling needs an exposition of principles for ontology development. In this section some basic ideas are outlined.

14.7.1 Domains and Conceptualizations

The starting point of ontology development is a domain. The formation and emergence of domains is a result of the evolution of scientific knowledge about the world, but also the outcome of common sense reasoning and social knowledge as well as the result of philosophical contemplations and reflections. Ontological levels and strata, for example, as elaborated and exposed by the philosopher Hartmann (1964, 1965) and further developed by Poli in (2001), exhibit very comprehensive domains of the world. Another group of domains is related to the evolution of empirical sciences including, for example, the physical domain, the biological domain, the domain of medicine and many others. The domain of mathematics has a special nature and differs significantly from the domains of the empirical sciences. The realm of mathematics is the world of sets that are understood and conceived by the majority of mathematicians as platonic ideal entities which exhibit a subject-independent and atemporal existence.

A domain $D = (\text{Obj}(D), V(D), \text{CP}(D))$ is determined by a set $\text{Obj}(D)$ of objects associated to it, by a set V of views at $\text{Obj}(D)$, and by a set CP of classification principles for $\text{Obj}(D)$. The notion of view is used in an informal, intuitive sense, whereas the classification principle CP can be made usually more precise. In understanding, acquiring and representing the knowledge about a domain D we use categories and relations between them, and must specify the domain's objects and their fine-structure. Hence, we associate to a domain D two further constituents: the categories of D , denoted by $\text{Cat}(D)$, the relations of D , denoted by $\text{Rel}(D)$. These additional constituents are influenced by the view V at $\text{Obj}(D)$ and the classification principle CP of D . The system $\text{Concept}(D) = (\text{Obj}(D), \text{Cat}(D), \text{Rel}(D))$ can be conceived as a detailed form of a conceptualization of the domain D in the sense of (Gruber, 1993). This approach to conceptualizations supports the ideas expounded in (Mc Cray, 2006), since we assume that the categorial system $\text{Cat}(D)$ as well as the relations in $\text{Rel}(D)$, and also the classification principles $\text{CP}(D)$, depend on the deeper rooted world view of its designer including the purpose for which the categorial system is generated. An ontology of a domain D is based on a conceptualization $\text{Concept}(D)$; it is determined by adding axioms describing inter-relations between the categories and properties of relations. We present an ontology Ont by a system $\text{Ont} = (\text{Concept}(D), \text{Ax}(\text{Concept}(D)))$, where $\text{Ax}(\text{Concept}(D))$ denotes the set of axioms about the conceptualization $\text{Concept}(D)$. The most simple axioms are presented by relational links between categories.

The categories of $\text{Cat}(D)$ are divided into the set of principal categories of D , denoted by $\text{PrincCat}(D)$ of D , into the set of elementary categories of D , designated by $\text{ElemCat}(D)$, into the set of aspectual categories of D , symbolized by $\text{AspCat}(D)$, and into the linguistically defined categories, denoted by $\text{LingCat}(D)$. These sets of categories form an increasing chain, i.e. we suppose that $\text{PrincCat}(D) \subseteq \text{ElemCat}(D) \subseteq \text{AspCat}(D) \subseteq \text{LingCat}(D)$. The principal categories are the most fundamental ones of a domain. For the domain of biology the category of organism is accepted as being principal. The linguistically defined categories are introduced by formulas of a language, usually a formal language L . The system $(\text{PrincCat}(D), \text{ElemCat}(D), \text{AspCat}(D))$ is called a graduated conceptualization of D .

The elementary categories of a domain are introduced and determined by a classification based on the domain's classification principles; they usually present a taxonomy. There is a great variety of combinations of the classification principles being applied to specify elementary categories. A domain D is called simple if it has only one view and one classification principle, and if the taxonomy based on $(\text{Ind}, \text{V}, \text{CP})$ exhibits a tree-like structure. If the domain has multiple views then the taxonomic ordering of elementary concepts cannot be assumed to be tree-like. Multiple views can be the reason for the occurrence of multiple inheritance. Aspectual categories are derived from elementary categories by aspectual composition and deployment.

14.7.2 Steps of Ontology Development

We summarize the basic steps for the development of an ontology. An ontology usually is associated to a domain, hence, we must gain an understanding of the domain which is under consideration. The constituents of a domain D include the objects of D , the assumed views V , and the classification principles to be used for the construction of concepts. These constituents can be analysed in the framework of a top level ontology. We sketch a top-level centred approach to ontology development and use as a basis the ontology GFO.

1. Step: Domain Specification and Proto-Ontology. A domain is determined by classification principles and a set of views. The first step is the construction of a domain specification. In particular, a description of the objects of the domain A must be established. The considered objects are determined by the assumed views, whereas the classification principles provide the means for structuring the set $\text{Obj}(D)$ of objects. Usually, there is source information which is associated to the domain, in particular a set $\text{Terms}(D)$ of terms denoting concepts in the domain. The system $\text{ProtoOnt}(D) = (\text{Spec}(D), \text{Terms}(D))$ consisting of the domain specification $\text{Spec}(D)$ and a set of terms $\text{Terms}(D)$ is called a *proto-ontology*. A proto-ontology of a domain contains the relevant information needed to make the further steps in developing a *axiomatized ontology* about D .

2. Step: Conceptualisation. A conceptualisation is based on a proto-ontology; the result of this step is a *graduated conceptualization*. Hence, the principal and elementary concepts of the domain must be identified or introduced. The resulting

concepts belong either to the concepts denoted by the terms of $\text{Terms}(D)$ or they are constructed by means of the classification principles. A further sub-step is pertained to the desired aspectual concepts which are derived from the elementary concepts. Finally, we must identify relations which are relevant to capture content about the individuals and concepts. It would be helpful if a meta-classification of relations is available. GFO provides already a basic classification of relations which must be extended and adapted to the particular domain D .

3. Step: Axiomatisation. During this step axioms are developed. This needs a formalism, which can be a graph-structure or a formal language. We expound in more detail the construction of a formal knowledge bases assisted and supported by a top-level ontology TO. Generally, a axiomatized ontology $\text{Ont} = (L, V, \text{Ax}(V))$ consists of a structured vocabulary V , called ontological signature, which contains symbols denoting categories, individuals, and relations between categories or between their instances, and a set of axioms $\text{Ax}(V)$ which are expressions of the formal language L . The set $\text{Ax}(V)$ of axioms captures the meaning of the symbols of V implicitly. A definitional extension $\text{Ont}^d = (L, V \cup C(DF), \text{Ax}(V) \cup DF)$ of Ont is given by a set DF of explicit definitions over the signature V and a new set $C(DF)$ symbols introduced by the definitions. Every explicit definition has the form $t := e(V)$, where $e(V)$ is an expression of L using only symbols from V (hence the symbol t does not occur in $e(V)$).

An *ontological mapping* M of a conceptualization $\text{Conc}(D)$ into an axiomatized ontology Ont is given by a pair $M = (tr, DF)$ consisting of a definitional extension Ont^d of Ont by (the set of definitions) DF and by function tr which satisfies the following condition:

For every term $t \in \text{Tm}$ denoting a concept C of $\text{Conc}(D)$ which is defined by the (natural language) expression $\text{Def}(t)$ the function tr determines an expression $tr(\text{Def}(t))$ of the extended language $L(V \cup C(DF))$ such that $\text{Def}(t)$ and $tr(\text{Def}(t))$ are semantically equivalent with respect to the knowledge base $\text{Ax}(\text{Ont}) \cup DF$.

Then the set $\text{OntMap}(\text{Conc}(D)) = \text{Ax}(V) \cup DF \cup \{tr(\text{Def}(t)) : t \in \text{Conc}(D)\}$ is a formal knowledge base which formally captures the semantics of the conceptualization of D . The notion of *semantical equivalence with respect to a knowledge base* is used here informally because a strict formal semantics for natural language sentences does not yet exist; the notion has to be read “the meaning of the natural language (or semi-formal) sentence $\text{Def}(t)$ is equivalent to the meaning of the expression $tr(\text{Def}(t))$ ”. An expression e is considered as ontologically founded on an ontology Ont if it is expressed in some definitional extension Ont^d of Ont . Hence, an ontological mapping of a conceptualisation $\text{Conc}(D)$ associates to every term of $\text{Conc}(D)$ an equivalent formal description which is based on a formally axiomatized ontology Ont . A final axiomatization for $\text{Conc}(D)$ can be achieved by starting with a top-level ontology, say GFO, and then constructing by iterated steps an ontological mapping from $\text{Conc}(D)$ into a suitable extension of GFO. An advanced elaboration of this theory, which is being investigated by the Onto-Med group, is presented in Herre and Heller (2006a). The construction of an ontological mapping, which yields an axiomatization of the conceptualization includes, according to Herre and Heller (2006a), three main tasks:

1. Construction of a set PCR of primitive concepts and relations out from the set $\{Def(t) : t \in Conc\}$ (*problem of primitive basis*)
2. Construction of an extension TO_1 of TO by adding new categories Cat and relations Rel and a set of new axioms. $Ax(Cat \cup Rel)$ (*axiomatizability problem*)
3. Construction of equivalent expressions for $Def(t) \cup PCR$ on the base of TO_1 (*definability problem*).

14.7.3 Ontological Modelling

In this section we set forth some ideas on a new area of research which aims at the use of ontologies for modelling of processes and their simulations. The basic idea of ontological modelling is expounded in Herre and Heller (2006a). Subsequently, we add further ideas on this field of research. The starting point is a class P of (natural) processes which can be considered to be included in a class of situoids. A strict ontological model of P is a category $C := \text{OntMod}(P)$ whose extension equals P , i.e. for all processes p holds: $p \in P$ if and only if $p :: C$. Usually, the condition of a strict ontological model for a class P of processes can hardly be achieved. For this reason we say that C is an ontological model of P if the extension of P sufficiently approximates the class P .³¹ A process $p \in P$ is said to be computable if there is an execution of an algorithm α which approximates p .³² The class P of processes is said to be computable if there is an algorithm α such that every $p \in P$ is approximated by an execution of α . We assume that any reasonable class P of natural processes is computable.³³

The paper Roeder and Loeffler (2002) expounds a conceptual analysis of the notion of a stem cell. The authors argue that stemness of a cell cannot be considered as a specific property that can be determined at one time-point without putting the cell to functional tests. Hence, stem cells exhibit stemness by participating in certain interacting processes which are embedded in a larger process which we call stem-cell process. Let SCP be the class of all stem-cell processes. An expressive ontological model for SCP should be specified in a formal language which includes among others the conditions presented in the model description in Roeder and Loeffler (2002). But, more properties must be taken into consideration, among them those which pertain to different granularity levels of the stem cell processes, but also to properties which are related to the snapshots of the process and its sub-processes. Hence, the acquisition of relevant presentalist and processual properties

³¹This vagueness cannot be avoided because we assume that the specification of $\text{OntMod}(P)$ exhibits a decidable set of conditions. By Gödel's incompleteness theorems a complete specification of P cannot be, in general, achieved.

³²The term "an execution of α approximates p " needs further explanation. This can be made precise by using the approaches of computable and constructive analysis (Weihrauch, 2000; Geuvers et al., 2007). The development of an ontological theory of computational simulation of natural processes is in progress and will be published elsewhere.

³³It is an open problem whether every reasonable natural process is computable (Kreisel, 1974).

of stem cell processes is important. Using this idea of ontological models a number of notions can be clarified and refined, among them computer simulation, prediction, practical experiment, etc.

Acknowledgments Many thanks to P. Burek, R. Hoehndorf, F. Loebe, H. Michalek who contributed significantly to the development of GFO. I am grateful to R. Poli and anonymous reviewers for their critical remarks that contribute to the quality of the paper. Thanks to M. West for fruitful discussions which lead to deeper insight into 4-dimensionalism. Finally, thanks to J. Gracia for inspiring discussions on the relations between different kinds of categories, and the proper interpretation of the notion of realism.

References

- Allen, J.F., and P.J. Hayes. 1990. Moments and points in an interval-based temporal logic. *Computational Intelligence* 5(4):225–238.
- Armstrong, D.M. 1997. *A world of states of affairs*. New York, NY: Cambridge University Press.
- Ashburner, M., C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, et al. 2000. Gene ontology: Tool for the unification of biology. *Nature Genetics* 25(1):25–29.
- Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge, UK: Cambridge University Press.
- Bachman, C.W., and M. Daya. 1977. The role concept in data models. In Proceedings of the 3rd International Conference on Very Large Databases, 464–476. IEEE Computer Society.
- Bard, J., S.Y. Rhee, and M. Ashburner. 2005. An ontology for cell types. *Genome Biology* 6(2):R21.
- Barwise, J., and J. Perry. 1983. *Situations and Attitudes*. Cambridge, MA: Bradford Book/MIT Press.
- Bloch, E. 1985. *Subject-Objekt*. Frankfurt/M: Suhrkamp.
- Boella, G, J. Odell, L. van der Torre, and H. Verhagen, eds. 2005. Proceedings of the 2005 AAAI Fall Symposium 'Roles, an Interdisciplinary Perspective: Ontologies, Languages, and Multiagent Systems', Nov 3–6, Arlington, Virginia, number FS-05-08 in Fall Symposium Series Technical Reports, Menlo Park, CA: AAAI Press.
- Booch, G., J. Rumbaugh, and I. Jacobson. 1999. *The unified modeling language user guide*. Object Technology Series. Reading, MA: Addison Wesley.
- Braunwald, E., K.J. Isselbacher, R.G. Petersdorf, J.D. Wilson, J.B. Martin, and A.S. Fauci, eds. 1987. *Harrison's principles of internal medicine*, 11th edition. New York, NY: McGraw-Hill.
- Brentano, F. 1976. *Philosophische Untersuchungen zu Raum, Zeit und Kontinuum*, eds. S. Körner, and R.M. Chisholm, Hamburg: Felix- Meiner Verlag.
- Brooksbank, C., G. Cameron, and J. Thornton. 2005. The european bioinformatics institute's data resources: Towards systems biology. *Nucleic Acids Research* 33(Database issue):D46–D53.
- Burek, P. 2007. Ontology of functions: A domain-independent framework for modeling functions, PhD thesis. University of Leipzig, Institute of Informatics (IfI).
- Burek, P., R. Hoehndorf, F. Loebe, J. Visagie, H. Herre, and J. Kelso. 2006. A top-level ontology of functions and its application in the open biomedical ontologies. *Bioinformatics* 22(14):e66–e73.
- Casati, R., and A. Varzi. 1994. *Holes and other superficialities*. Cambridge, MA: MIT Press.
- Chandrasekaran, B., and J.R. Josephson. 1997. Representing functions as effect. In Proceedings of the Functional Modeling Workshop, Paris, France.
- Chisholm, R.M. 1983. Boundaries as dependent particulars. *Grazer Philosophische Studien* 20:87–96.
- Chang, C.C., and H.J. Keisler. 1977. *Model theory*. Amsterdam: North-Holland Publishing Company.

- de Keizer, N.F., A. Abu-Hanna, and J.H.M. Zwetsloot-Schonk. 2000. Understanding terminological systems I: Terminology and typology. *Methods of Information in Medicine* 39(1):16–21.
- Dori, D. 2002. *Object-process methodology: A holistic systems paradigm*. Berlin: Springer.
- Gärdenfors, P. 2000. *Conceptual spaces: The geometry of thought*, A Bradford Book. Cambridge, MA: MIT Press.
- Genesereth, M.R., and R.E. Fikes. 1992. Knowledge interchange format. Technical Report Logic-92-1, Stanford Logic Group, Stanford.
- Geuvers, H., M. Niggui, B. Spitters, and F. Wiedijk. 2007. Constructive analysis, types and exact real numbers. *Mathematical Structures in Computer Science* 17:3–36.
- Gracia, J.J.E. 1999. *Metaphysics and its tasks: The search for the categorial foundation of knowledge*. SUNY series in Philosophy. Albany, NY: State University of New York Press.
- Grenon, P. 2003. Spatio-temporality in basic formal ontology: SNAP and SPAN. http://www.ifomis.org/Research/IFOMISReports/IFOMIS%20Report%2005_2003.pdf
- Gruber, T.R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2):199–220.
- Guarino, N. 1992. Concepts, attributes and arbitrary relations: Some linguistic and ontological criteria for structuring knowledge bases. *Data & Knowledge Engineering* 8(3): 249–261.
- Guarino, N., and C.A. Welty. 2000. A formal ontology of properties. In *Knowledge engineering and knowledge management: methods, models, and tools*, eds. R. Dieng and O. Corby. Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW2000), Juan-les-Pins, France, Oct 2–6, LNCS 1937, 97–112. Berlin: Springer.
- Guarino, N., and C.A. Welty. 2001. Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering* 39(1):51–74.
- Guarino, N., and C. Welty. 2004. An overview of ontoclean. In *Handbook on ontologies*, International Handbooks on Information Systems, Chapter 8, eds. S. Staab, and R. Studer, 151–159. Berlin: Springer.
- Guizzardi, G., and G. Wagner. 2004a. Towards ontological foundations for agent modelling concepts using the unified foundational ontology (UFO), LNCS 3508, June 2004, 110–124. doi:<http://dx.doi.org/10.1007/1147248>
- Guizzardi, G., and G. Wagner. 2004b. A unified foundational ontology and some applications of it in business modelling. Proceedings of the CAiSE'04 Workshops. Faculty of Computer Science and Information Technology, Riga Technical University, Riga, June 2004, vol. 3, 129–143, Riga, Latvia.
- Hartmann, N. 1964. *Der Aufbau der realen Welt*. Berlin: Walter de Gruyter and Co.
- Hayes, P.J. 1995. A catalogue of temporal theories. Technical Report UIUC-BI-AI-96-01, University of Illinois.
- Heller, B., H. Herre, K. Lippoldt, M. Löffler. 2004. Standardized terminology for clinical trial protocols based on ontological top-level categories. In *Computer-based support for clinical guidelines and protocols*, eds. K. Kaiser, S. Miksch, S.W. Tu. *Proceedings of the Symposium on Computerized Guidelines and Protocols*. (CGP 2004), 13–14 Apr 2004. Prague. 46–60. Studies in Health Technology and Informatics, vol. 101. Amsterdam: IOS-Press.
- Herre, H., and B. Heller. 2006a. Semantic foundations of medical information systems based on top-level ontologies. *Journal of Knowledge-Based Systems* 19(2):107–115.
- Herre, H., B. Heller[†], P. Burek, F. Loebe, R. Hoehndorf, and H. Michalek. 2006b. General formal ontology (GFO) – A foundational ontology integrating objects and processes, Report Nr. 8, Onto-Med, IMISE.
- Herre, H. 2010. Ontology of mereological systems – A logical approach, this volume.
- Hermes, H. 1959. Zur Axiomatisierung der Mechanik. In *The axiomatic method*, eds. L. Henkin, P. Suppes, and A. Tarski, 282–290. Amsterdam: North-Holland Publishing Company.
- Ingarden, R. 1964. *Der Streit um die Existenz der Welt I (Existentialontologie)*. Tübingen: Max Niemeyer Verlag.

- Johansson, I. 1989. *Ontological investigations: An inquiry into the categories of nature, man and society*. New York, NY: Routledge.
- Johnston, M., and G. Forbes. 1987. Is there a problem about persistence? *Aristotelian Society* 61:107–135.
- Kreisel, G. 1974. A notion of mechanistic theory. *Synthese* 29:16–26.
- J. Lehmann, S. Borgo, A. Gangemi, and C. Masolo. 2004. Causality and causation in DOLCE. In Formal Ontology in Information Systems: Proceedings of the International Conference FOIS 2004, vol 114 of *Frontiers in Artificial Intelligence And Applications*, eds. A.C. Varzi, and L. Vieu, 273–284, Amsterdam: IOS Press.
- D.B. Lenat, and R.V. Guha. 1990. *Building large knowledge-based systems: Representation and inference in the cyc project*. Reading, MA: Addison-Wesley.
- Lewis, D. 1986. *On the plurality of worlds*. Oxford: Basil blackwell.
- Loebe, F. 2003. An analysis of roles: Towards ontology-based modelling. *Onto-Med Report* 6, Onto-Med Research Group, University of Leipzig.
- Loebe, F. 2003. Abstract vs. social roles: A refined top-level ontological analysis. In Boella et al. [8], 93–100.
- Loebe, F. 2007. Abstract vs social roles_ Towards a general theoretical account or roles. *Applied Ontology* 2(2):127–158.
- Loux, M. 1998. *Metaphysics: A contemporary introduction*. New York, NY: Routledge.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. 2003. Wonderweb deliverable D18: Ontology library (final). Technical Report, Laboratory for Applied Ontology – ISTC-CNR, Trento.
- Masolo, C., S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. 2002. Wonderweb deliverable D17. Preliminary Report Version 2.0, Laboratory for Applied Ontology – ISTC-CNR, Padova, IT.
- Masolo, C., L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. 2004. Social roles and their descriptions. In Principles of Knowledge Representation and Reasoning: Proceedings of the 9th International Conference (KR2004.), eds. D. Dubois, C. Welty, and M.-A. Williams, Whistler, Canada, June 2–5, 267–277, Menlo Park, CA: AAAI Press.
- MC Cray, A.T. 2006. Conceptualizing the world: Lessons from history. *Journal of Biomedical Informatics* 39:267–273.
- Niles, I., and A. Pease. 2001. Towards a standard upper ontology. In Formal Ontology in Information Systems: Collected Papers from the 2nd International Conference, Oct 2001, eds. C. Welty, and B. Smith, 2–9, New York, NY: ACM Press.
- Pease, A., and I. Niles. 2002. IEEE standard upper ontology: A progress report. *Knowledge Engineering Review* 17(1):65–70.
- Poli, R. 2001. The basic problem of the theory of levels of reality. *Axiomathes* 12(3–4):261–283.
- Poli, R. 2002. Ontological methodology. *International Journal of Human-Computer Studies* 56(6):639–664.
- Pschyrembel, W., and O. Dornblüth. 2002. In *Pschyrembel Klinisches Wörterbuch*. 259th edition, Berlin: Walter de Gruyter.
- Ridder, L. 2002. *Mereologie*. Frankfurt a. Main: Vittorio Klostermann.
- Roeder, I., and M. Loeffler. 2002. Anoval dynamic model of hematopoietic stem cell organizations based on the concept of within-tissue plasticity. *Experimental Hematology* 30: 853–861.
- Rosch, E. 1975. Cognitive representations of semantic categories. *Journal of Experimental Psychology*, 104:192–233.
- Rosse, C., and J.L. Mejno. 2003. A reference ontology for biomedical informatics. The foundational model of anatomy. *Journal of biomedical informatics* 36:478–500.
- Rumbaugh, J., I. Jacobson, and G. Booch. 1999. *The unified modeling. language reference manual*. Object Technology Series. Reading, MA: Addison Wesley.
- Russell, S., and P. Norvig. 1995. *Artificial intelligence: A modern approach*. Prentice hall series in artificial intelligence. Upper Saddle River, NJ: Prentice Hall.

- Sasajima, M., Y. Kitamura, M. Ikeda, and R. Mizoguchi. 1995. FBRL: A function and behavior representation language. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 95, Montreal, Quebec, Canada, Aug 20–25, 1830–1836. San Francisco, CA: Morgan Kaufmann.
- Searle, J.R. 1995. *The construction of social reality*. New York, NY: Free Press.
- Seibt, J. 2003. Free process theory: towards a typology of processes. *Axiomathes*. 14(1):23–55.
- Shahar, Y. 1994. A knowledge-based method for temporal abstraction of clinical data. PhD thesis, Stanford, CA: Stanford University.
- Sider, T. 2001. *Four-dimensionalism: an ontology of persistences and time*. Oxford: Clarendon Press.
- Smith, B., and A. Varzi. 2000. Fiat and bona fide boundaries. *Philosophy and Phenomenological Research* 60(2):401–420.
- Smith, B. 2004. Beyond concepts: Ontology as reality representation. In FOIS, International conference on formal ontology and information systems, 73–84, Turin: IOS Press.
- Smith, B., W.Ceusters, and R. Temmermann. 2005. Wüsteria. In Proceedings Medical Informatics Europe 2005, Geneva; *Studies in Health Technology and Informatics* 116:647–652.
- Smith, B. 2006. From concepts to clinical reality: An essay on the benchmarking of biomedical terminologies. *Journal of Biomedical Informatics* 39:288–298.
- Smith, B., and W. Ceusters. 2006. Ontology as the core discipline of biomedical informatics. In *Computing, philosophy, and cognitive science*. eds. C.D. Crnkovic, and S. Stuart. Cambridge, MA: Scholars Press.
- Smith, B. et. al. 2007. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology* 25(11):1251–1255, Nov 2007.
- Sowa, J.F. 1984. *Conceptual structures: Information processing in mind and machine*. Reading, MA: Addison-Wesley.
- Sowa, J.F. 2000. *Knowledge representation: logical, philosophical and computational foundations*. Pacific Grove, CA: Brooks/Cole.
- Steimann, F. 2000. On the representation of roles in object-oriented and conceptual modelling. *Data & Knowledge Engineering* 35(1):83–106.
- Sunagawa, E., K. Kozaki, Y. Kitamura, and R. Mizoguchi. 2005. A framework for organizing role concepts in ontology development tool: Hozo. In Boella et al. [8], 136–143.
- SUO. 2004. IEEE P1600.1 Standard upper ontology working group (SUO WG). <http://suo.ieee.org>.
- Szczerba, L.W. 1977. Interpretability of elementary theories. In eds. R.E. Butts, and J. Hintikka, *Logic, foundations of mathematics and computability theory*, volume 9 of *The Western Ontario Series in Philosophy of Science*, 129–145. Dordrecht: Reidel.
- Tarski, A. 1944. The semantic conception of truth and the foundation of semantics. *Philosophy and Phenomenological Research* 4:341–375.
- Tarski, A. 1983. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*, 2nd edition., eds. J. Corcoran, Indianapolis: Hackett.
- W3C. 2004. Web Ontology Language (OWL) Specifications. W3C Recommendations, World WideWeb Consortium (W3C), Cambridge, MA. <http://www.w3.org/2004/OWL/>.
- Weihrauch, K. 2000. *Computable analysis*. Berlin: Springer.
- Wertheimer, M. 1912. Experimentelle Studien über das Sehen von Bewegung. *Zeitschrift für Psychologie* 161–265.
- Wertheimer, M. 1922. Untersuchungen zur Lehre von der Gestalt. I. *Prinzipielle Bemerkungen*.
- West, M., J. Sullivan, and H. Teijgeler. 2003. ISO/FDIS 15926-2: Lifecycle integration of process plant data including oil and gas production facilities. ISO TC184/SC4/WG3N1328, July 2003. http://www.tc184-sc4.org/wg3ndocs/wg3n1328/lifecycle_integration_schema.html.
- Wilkerson, T.E. 1995. *Natural kinds*. Avebury Series in Philosophy. Aldershot, UK: Avebury.
- Wittgenstein, L. 1922. *Tractatus logico-philosophicus*. (transl: C.K. Ogden) London: Routledge & Kegan Paul.

Chapter 15

Ontologies in Biology

Janet Kelso, Robert Hoehndorf, and Kay Prüfer

15.1 Introduction

Modern biology is a data-producing, data-driven science. Biological databases covering the domains of sequence, structure, phenotype, and many other types of biological information are core resources for biomedical research. Recent advances in molecular biology, coupled with rapid development of high-throughput technologies, have led to the exponential growth of databases housing information about the sequences, functions and localizations of genes and proteins for a wide range of organisms. The bottleneck is therefore no longer the production of data, but the integration and analysis of this data. In order to make biologically meaningful discoveries, researchers require the ability to query and extract the biological information available from a variety of sources, and to integrate this information in meaningful ways. However, there are a number of obstacles that make such integrated analyses difficult.

- (i) With the exception of the major nucleotide and protein databases biological databases are generally developed and maintained by the community of scientists that are interested in the scientific questions that can be addressed by the data being stored in the database. As such it is common that biological data is stored in geographically disparate locations, using different technologies and representations. Redundancy or partial overlap in stored data is common.
- (ii) The integration of biological data has been severely hindered by ambiguities in terminology, semantics and storage. Synonyms and abbreviations are widely used and often applied with conflicting meanings. Homonyms are present both within and between biological sub-disciplines. Further, the definitions of

J. Kelso (✉)

Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, Leipzig 04103, Germany
e-mail: kelso@eva.mpg.de

R. Hoehndorf (✉)

Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, Leipzig 04103, Germany;
Department of Computer Science, University of Leipzig, Leipzig, Germany
e-mail: leechuck@leechuck.de

even fundamental biological concepts such as *organism*, *gene* and *species* are not universally agreed upon. It is likely that these problems have arisen as a side-effect of our constantly evolving understanding of biological systems, and as a result of the gradual merging of historically distinct sub-disciplines as biological research becomes more integrative.

In 1998 Steffen Schulze-Kremer presented a paper at the Pacific Symposium of Biocomputing (Schulze-Kremer, 1998) in which he discussed the application and potential future applications of ontologies in molecular biology. Both in this paper, and in a later paper (Schulze-Kremer, 2002), he clearly identified the information exchange and data integration problems prevalent in the biological sciences saying: “Many researchers and databases use (at least partially) idiosyncratic terms and concepts for representing biological information. Often, terms and definitions differ between groups, with different groups not infrequently using identical terms with different meanings. The concept ‘gene’, for example, is used with different semantics by the major international genomic databases.” He proposed ontologies as a means to provide standardized nomenclature for the rapidly growing databases of sequence, structure, expression, metabolic and regulatory data for many organisms.

Recent years have seen a growing trend towards the development and adoption of ontologies for the management of biological knowledge. Ontologies and controlled vocabularies for various domains of the biomedical sciences have been developed, largely in an effort to provide a shared language for communicating biological information. Ontologies are viewed by the biomedical community as a powerful means to represent, analyze and integrate biological information.

More historically, however, much of the original basis of biology is in the classification of domains. An early example of the classification of organisms are the taxonomies formulated by Linnaeus. The controlled vocabulary of MeSH terms used by Entrez at the NCBI portal¹ of the National Library of Medicine are another example of where a structured set of terms are used to classify publications and index them for searching.

From a biologist’s perspective, a controlled terminology with structured relationships is useful in many domains. It provides a consistent and defined nomenclature and provides structured access to possible terms and relationships.

The major recent utilisation of ontologies in biomedicine has been largely to provide a common terminology for a variety of domains (discussed later in this chapter). Successful utilisation of ontologies is dependent upon multiple factors including their usability, design and on their broad adoption by the community. There has been some debate over whether a single all-encompassing ontology or smaller domain or task-specific ontologies are more useful.² Smaller ontologies take less time to build and are simpler to maintain and grow. As a result of their practicality, smaller ontologies relevant to distinct domains of molecular biology have been rapidly developed and put to use. In order for ontologies in a domain to be

¹<http://www.ncbi.nlm.nih.gov/>

²[http://en.wikipedia.org/wiki/Upper_ontology_\(computer_science\)](http://en.wikipedia.org/wiki/Upper_ontology_(computer_science))

accepted as the standard, community involvement and adoption are essential and this community agreement has been a hallmark of the development of the modern “bio-ontologies”. Many of the widely used and accepted ontologies have been built by consortia, and are designed with specific applications in mind. The most common application of these bio-ontologies has been the formalisation of domains of biomedical knowledge through explicit and unambiguous definition of terms used for the description of biological data. This has been achieved through the naming and definition of relevant entities within biomedical domains, and the specification of the relationships that exist between them. Additionally, ontologies that specify the schemas of knowledge bases have also been valuable in providing a basis for a variety of standards specifications for the collection of gene expression (Whetzel et al., 2006), and sequence data (Field et al., 2006).

In this chapter we will explore a few of the modern bio-ontologies and will discuss their scope, strengths and weaknesses. We will include a short description of the resources and applications that have been developed around these bio-ontologies with a view to showing how valuable these ontologies have become in supporting biological research. The application of formal ontological principles to the design of many of the biological ontologies has lagged behind the development of “light-weight” domain ontologies, and as a result the scope of applications remains restricted. We will discuss some of the criticism of the lack of formality in the bio-ontologies, and provide some ideas about how formal logics can be used to address the growing need for ontology integration.

15.2 Ontologies in Biomedicine

The use of ontology in biomedicine has a long history. Some of the older medical terminological systems are discussed in some detail in [Chapter 16](#) by Herre, this volume. We will not discuss these in any detail, but will focus on the second generation of biomedical ontologies which appeared in recent years.

Growth in the development and use of ontologies in biology in the last 10 years has been driven by the need for biologists to organise large volumes of data being generated in molecular biology. To share this data effectively it was necessary to identify and agree on the relevant concepts and select a shared set of terms for the description of these domains. Based on this early start a large and growing number of bio-ontologies have arisen.

15.2.1 *The Open Biomedical Ontologies*

The Open Biomedical Ontologies (OBO) project is an umbrella organization which hosts a library of ontologies for the biomedical domain.

To be included in the OBO, ontologies need to conform to a set of criteria design which ensure their quality and inter-operability. The OBO co-ordinators provide guidelines for ontology development and facilitate communication between the ontology developers in order to support the development of such ontologies.

The OBO Foundry,³ a project attempting to increase the formal rigour of the OBO ontologies through the application ontological principles, is based on the following set of principles:

- The ontology must be *open* and available to be used by all without any constraint other than (a) its origin must be acknowledged and (b) it is not to be altered and subsequently redistributed under the original name or with the same identifiers.

Making the ontologies freely available is intended to increase acceptance and use of the ontology, which in turn ensures that the content is accurate and reflects the views of the community.

- The ontology is in, or can be expressed in, a common shared syntax. This may be either the OBO syntax, extensions of this syntax, or OWL.

The motivation for this principle is that it aids in facilitating inter-operability and permits the development of tools and methods which can then be usefully applied to multiple domains.

- The ontology possesses a unique identifier space within the OBO Foundry.
- The ontology provider has procedures for identifying distinct successive versions.
- The ontology has a clearly specified and clearly delineated content. The ontology must be orthogonal to other ontologies already lodged within OBO.

The major reason for this principle is to allow two different ontologies, for example anatomy and process, to be combined through additional relationships. These relationships could then be used to constrain when terms could be jointly applied to describe complementary (but distinguishable) perspectives on the same biological or medical entity. As a corollary to this, the OBO Foundry strives for community acceptance of a single ontology for one domain, rather than encouraging rivalry between ontologies.

- The ontologies include textual definitions for all terms.
- The ontology uses relations which are unambiguously defined following the pattern of definitions laid down in the OBO Relation Ontology.
- The ontology is well documented.
- The ontology has a plurality of independent users.
- The ontology will be developed collaboratively with other OBO Foundry members.

A wide variety of biomedical domains are covered by the OBO including ontologies of anatomy, development and disease for a number of key organisms, ontologies of biological sequence, function and process, and ontologies of biochemistry, cell types and behaviour.

Here we discuss the Gene Ontology as an example of a successful and widely used biomedical ontology which forms part of the Open Biomedical Ontology Foundry collection.⁴ More than 65 additional ontologies in various domains and stages of development are included in the OBO Foundry. Some of the key ontologies are described in Table 15.1.

³<http://www.obofoundry.org/>

⁴<http://www.obofoundry.org/>

Table 15.1 Ontologies that form part of the OBO Foundry. There are many more ontologies that are part of the OBO library. These cover a broad range of domains including the anatomies and development of various eukaryotic organisms (plant and animal), disease, behaviour

Domain	Ontology name	Description	
Molecular biology	Gene ontology	GO	Three ontologies which describe the molecular function, cellular location and biological process for genes and gene products
Molecular biology	Sequence ontology	SO	Ontology for biological sequence annotation and the description of sequence objects in databases
Organism anatomy and cell type	Common anatomy reference ontology	CARO	A common template for anatomical ontologies
Organism anatomy and cell type	Cell ontology	CL	A controlled vocabulary of cell types. Not organism-specific
Organism anatomy and cell type	Foundational Model of Anatomy	FMA	A reduced version of the FMA with only the <i>is_a</i> , <i>part_of</i> and <i>has_part</i> relations included.
Phenotypic qualities	Phenotype ontology	PATO	Ontology of phenotypic qualities/properties intended for use with other ontologies such as organism-specific anatomies.
Biochemical	Chemicals of biological interest	CheBI	Classification of the chemicals with relevance in the biological domain. Can be used in conjunction with other ontologies.
Experimental measurement	Ontology for Biomedical Investigation	OBI	Ontology for describing the design of an experiment, the protocols, materials and instrumentation used, the data generated and the type of analyses performed
Experimental measurement Relations	Unit Relationship ontology	UNIT RO	Metric unit ontology intended for use with PATO Ontology of the relations being used by the OBO ontologies

15.2.2 *The Gene Ontology*

The Gene Ontology (Ashburner et al., 2000) provides three structured, controlled, non-organism specific vocabularies describing the entities that exist in the domains of molecular function, cellular location and biological processes of genes or gene products.

The project began in 1998 as a collaboration between the curators of three of the major model organism databases (FlyBase, the Mouse Genome Informatics database, and the Saccharomyces Genome Database), and arose out of the need for these communities to share a common, unambiguous vocabulary for functional annotation of genes and gene products within these databases.

The aims of the Gene Ontology consortium, which has since expanded to include 16 members, are: (i) to develop a set of controlled, structured vocabularies to describe key domains of molecular biology, including gene product attributes and biological sequences; (ii) to apply GO terms in the annotation of sequences, genes or gene products in biological databases; and (iii) to provide a centralized public resource allowing universal access to the ontologies, annotation data sets and software tools developed for use with GO data. (Harris et al., 2004). The success of the GO is evidenced by its widespread adoption. Using the search term “Gene Ontology” identifies more than 1,843 citations in GoogleScholar in June 2007. It was the success of the Gene Ontology that inspired the development of a large number of domain ontologies, many of which are now gathered under the umbrella of the OBO consortium. In understanding the reasons for this success it is important to note that the GO consortium focused on openness and community-involvement, and the application to real data as key principles in the development, and that these, together with others factors discussed in an opinion article (Lewis, 2005) have proven extremely powerful motivators for the biomedical community.

The entities captured in each of the three ontologies that compose the Gene Ontology have *is-a* and *part-of* relations to other entities (Fig. 15.1). There is no explicit link between the three ontologies that make up the Gene Ontology, although relationships between the three ontologies exist. There have been various approaches to making these relationships explicit (Bodenreider et al., 2003; Bada and Hunter, 2007). The three ontologies are generally represented as directed acyclic graphs, so that multiple inheritance is possible. A large number of the other OBO ontologies are also represented as DAGs.

15.2.3 *Ontology Representation*

Many biomedical ontologies are made available in the OBO flatfile format (Golbreich and Horrocks, 2007). The OBO flatfile format in its original form specifies a directed acyclic graph (Fig. 15.2). In this graph, labeled nodes represent categories, labeled edges relationships between categories.

The Gene Ontology was the first ontology to use this representation format, together with one semantic rule, the True Path Rule. The true path rule states that

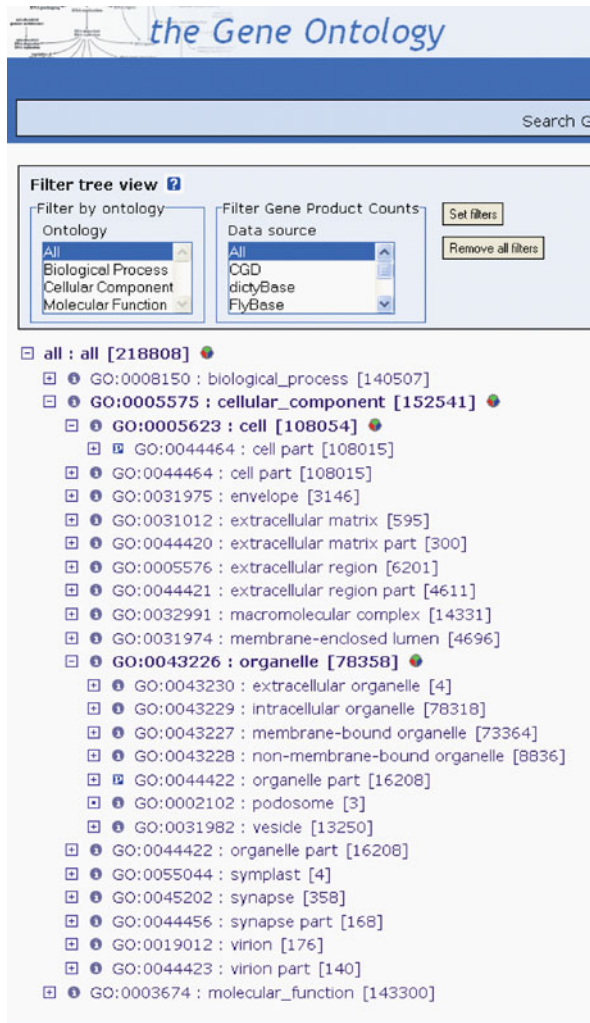
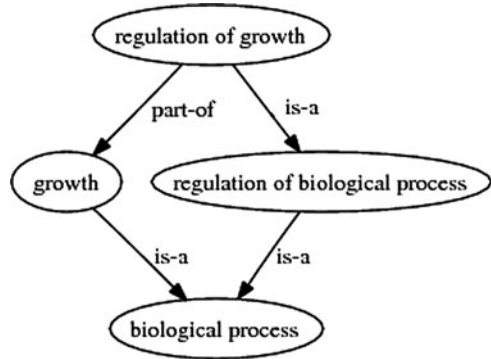


Fig. 15.1 A screenshot of a subsection of the Gene Ontology using the AmiGO browser.⁵ Relationship types are indicated to the left of the term accession as (I) for *is-a* or (P) for *part-of*. The *cellular compartment* ontology is expanded to show entities and their relationships. An additional feature is the ability to view genes and gene products which have been annotated with each term. The number in square brackets following the term name indicates how many gene products in public databases have been annotated using the term

“the pathway from a child term all the way up to its top-level parent(s) must always be true” (Ashburner et al., 2000). In the beginning, this rule was applied to the annotations of categories in the Gene Ontology: an annotation to a category remained

⁵<http://amigo.geneontology.org/>

Fig. 15.2 A part of the graph structure of the gene ontology. Nodes represent categories and edges are relations between the categories. The figure shows four categories, linked using *is-a* and *part-of* relations



a valid annotation for all *is-a* and *part-of* parents of the category. The annotation relation is not an ontological relationship and may have varying meanings. Therefore, a more precise definition and semantics for these DAGs was developed in first order logic (Smith et al., 2005) and description logics (Golbreich and Horrocks, 2007).

Let C be a set of concept names, $R \supseteq \{is - a\}$ a set of relationship names, $G = (V, E, c, r)$ be a labeled graph with vertices V , edges $E \subseteq V \times V$, a function $c: V \rightarrow C$ and a function $r: E \rightarrow R$. Then, G is equivalent to a theory T in first order logic over the signature $\Sigma = (\{:\} \cup R \cup C)$ such that for each $e \in E$:

- (1) If $r(e) = is - a$ and $e = (a, b)$ with $c(a) = c1$ and $c(b) = c2$, then $\{\forall x (x : c1 \rightarrow x : c2)\} \in T$
- (2) If $r(e) = S$ and $e = (a, b)$ with $c(a) = c1$ and $c(b) = c2$, then $\{\forall x (x : c1 \rightarrow \exists y (y : c2 \wedge S(x, y)))\} \in T$

The relationship “:” denotes the binary instantiation relation between an individual and a category. S denotes the additional relations used in this ontology. For example, if $R = \{part - of, is - a\}$ as in the Gene Ontology, an edge $e = (c1, c2)$ with the label $r(e) = part - of$ is translated to: for all $x (x : c1 \rightarrow \exists y (y : c2 \wedge part - of(x, y)))$.

One consequence of this definition and the True-Path Rule is that the *part-of* relation is transitive. Another important consequence is that the relation represented in the DAG is a necessary relation: there are no exceptions. The translation of a DAG into first order logic was not known from the beginning. Many of the criticism of the Gene Ontology and similar ontologies arose from misunderstandings of relations between categories. For a detailed discussion, see Sections 15.3, 15.4.1, and 15.4.4.

Recently, Semantic Web Technology is used for the development of biomedical ontologies. In particular, most ontologies that have commonly be represented in the OBO format as DAG are now available in OWL. Newly developed ontologies are often developed using a more expressive knowledge representation format, such as OWL.

15.2.4 Ontology Curation

Development and curation of the bio-ontologies is generally performed by domain experts, and consultation with ontologists is becoming more frequent. The OBO Foundry, for example, welcomes community input into OBO ontologies, and suggestions for changes or additions are implemented after careful evaluation by the curators. This process ensures that the ontology is a stable, versioned resource of high quality and consistency. Alternative models for ontology curation, including direct community curation via a wiki interface have been proposed (Hoehndorf et al., 2006) but are not yet widely adopted, largely due to concerns over the decrease in quality and increase in inconsistencies that may result if curation was completely unrestricted.

15.2.5 Annotation

A distinguishing feature of many biomedical ontologies is that they have been developed for specific use in the annotation of biomedical data such that this data can be shared and integrated. Annotation is the process whereby the terms from an ontology are associated with some experimental data (Fig. 15.3). For example, terms from the Gene Ontology have been used to describe the function, cellular location and biological process involvement of the genes and gene products in multiple model organism databases.

Annotations are contributed by consortium members and independent researchers. In the Gene Ontology the annotation data is generated largely by the collaborating model organism databases which then contribute these annotations to GO for storage and distribution. Each GO annotation has metadata identifying (i) who made the association between gene and GO term, (ii) the evidence supporting the association, and (iii) when the association was made.

Each association is labeled with an “evidence code” indicating the type of evidence that supports that association being made. Distinguishing between types of support for an association allows researchers using the data to decide how much confidence to place in the annotation. A large number of the annotations in the GO database are extracted from the biomedical literature by curators who read and interpret the statements about gene function and localization that are made in scientific papers. While manual curation provides the highest quality associations, it is time-consuming and dependent on skilled biologists. As a result high-throughput methods to associate annotations with genes/gene products using electronic methods have been developed. These approaches include extraction of associations from literature using text mining approaches, or the transfer of annotation from genes known to have similarity in their DNA sequence or protein structure. Direct experimental evidence confirmed by a human curator is generally considered more convincing than inference from automated analyses or associations based on sequence or structural similarities which have not been reviewed by a curator.

“Using microarray analysis, we identified RERG (ras-related and estrogen-regulated growth inhibitor). Like Ras, RERG protein exhibited intrinsic GDP/GTP binding and GTP hydrolysis activity. Unlike Ras proteins, RERG lacks a known recognition site for COOH-terminal prenylation and was localized primarily in the cytoplasm.”

Text string	GO Ontology	GO Term	GO ID
estrogen-regulated	process	response to hormone stimulus	GO:0009725
growth inhibitor	process	negative regulation of cell growth	GO:0030308
Ras, GDP/GTP binding	process	small GTPase mediated signal transduction	GO:0007624
GDP/GTP binding	function	GDP binding	GO:0019003
GDP/GTP binding	function	GTP binding	GO:0005525
GTP hydrolysis	function	GTPase activity	GO:0003924
cytoplasm	component	cytoplasm	GO:0005737

Fig. 15.3 An example of the process of annotating the protein RERG with terms from the gene ontology. Associations are made between the text of a scientific paper (*top*) and terms from the Gene Ontology biological process ontology (response to hormone stimulus, growth negative regulation of cell growth, small GTPase mediated signal transduction), molecular function ontology (GDP binding , GTP binding) and cellular component ontology (cytoplasm)

The genes/gene products from more than 35 distinct genomes have been annotated using the Gene Ontology. Additionally, the Gene Ontology Annotation (GOA) project⁶ (Camon et al., 2004) provides high quality GO-based annotations of the proteins in the UniProt knowledgebase. GOA provides annotated entries for over 60,000 species, making it the largest contributor the GO annotation effort. The annotations are generated through a combination of electronic and manual techniques. A list of all the available annotations can be retrieved from the GO project website.⁷ The various tools used to build the annotations are also distributed via the project website.⁸

A large number of the applications for which ontologies are used in biomedicine make extensive use of the annotations. These applications are discussed in more detail in Section 15.5 of this chapter.

⁶<http://www.ebi.ac.uk/GOA>

⁷<http://www.geneontology.org/GO.current.annotations.shtml>

⁸<http://www.geneontology.org/GO.tools.shtml#annot>

15.3 Criticism and Extension of the Gene Ontology

The Gene Ontology was criticized in a series of articles (Kumar et al., 2003; Smith et al., 2003; Kumar and Smith, 2004; Smith et al., 2004). Major confusion arose from the fact that, despite its name, the Gene Ontology is viewed by its curators as a controlled vocabulary rather than as an ontology. Important features of an ontology are missing from the Gene Ontology, most notably a formal specification and definition of the categories and relations in a formal language like description logic or first order logic. Although ontological notions such as *part*, *function*, *process*, and *object* are used in the names and textual definitions of the Gene Ontology's terms, none of these are properly defined.

The Gene Ontology has been further criticized for its lack of logical and ontological rigor. The representation as a directed acyclic graph was not formalized in the early stages of the project and the *part-of* relation was used in different, inconsistent ways within the ontologies. For example, organism-specific *part-of* statements were included in the Gene Ontology so that *part-of* statements were not always true, but only within the context of certain organisms.

These problems arose from two main areas. The first is a misunderstanding of the *part-of* relationship between categories. The definition of the *part-of* relationship was only added after a major part of the Gene Ontology had already been developed, with the result that *part-of* was not uniformly used according to the definition. In particular, default knowledge was included: *part-of* relations between categories which usually, but not always, hold true. The second reason for misunderstanding of *part-of* relations was the use of the same names for similar, but biologically unrelated types of entities in different organisms. An example is the fruiting body development in fungi and bacteria. In bacteria, the fruiting body development is a kind of cell communication, in fungi it is a kind of organ development. Whenever similar terms are used for different phenomena in different organisms, the Gene Ontology makes these terms organism specific by adding a "sensu" statement to the term. Therefore, fruiting body development (sensu Fungi) and fruiting body development (sensu Bacteria) are two different categories within the Gene Ontology.

Further analysis of the Gene Ontology revealed the implicit ontological distinctions made in its three disjoint taxonomic trees. There are three disjoint taxonomies: Cellular Component, Biological Process and Molecular Function. While cellular components are identified as subclasses of "substance", the distinction between Biological Process and Molecular Function proved to be more difficult. In particular, the Biological Process taxonomy contained terms such as "transport", while the Molecular Function taxonomy contained "transporter". In 2003, a major renaming of the terms in the Molecular Function taxonomy occurred, adding "activity" to the end of each term to reflect more closely the dynamic character of the terms described. However, the relationship between the Molecular Function taxonomy and the Biological Process taxonomy remained unclear, as did the exact nature of terms described in the Molecular Function taxonomy. The definition that relates Molecular Function to Biological Process is that a biological process is series of events accomplished by one or more ordered assemblies of molecular functions. This suggests

that the activities described in the Molecular Function taxonomy are a part of some biological process.

The analysis of the Gene Ontology according to the top-level ontological distinctions of the Basic Formal Ontology (BFO) (Grenon et al., 2004) concluded that cellular components are a subclass of the BFO’s continuant hierarchy, while biological processes and molecular functions as they are defined by GO are occurrents. It also concluded that molecular functions in GO are not a subclass of the function category in BFO, which are dependent continuants, but rather functionings.

A further analysis of the relationship between functions and processes in the Gene Ontology was performed using the Ontology of Functions (OF), a top-level ontology of functions (Burek et al., 2006). The OF provides a framework for defining the structure of functions, the function’s relation to processes and to objects

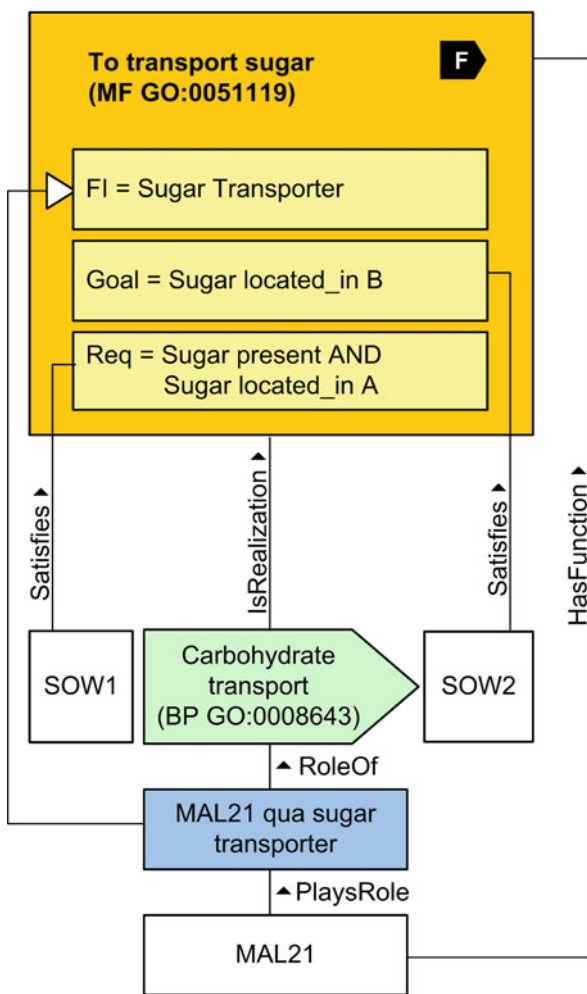


Fig. 15.4 The function “to transport sugar” represented in the framework of the ontology of functions. The function is represented using requirements and a goal. The functional item “sugar transporter” is the role that the function bearer (MAL21) plays in any realization of the function. The process “carbohydrate transport” (from gene ontology’s process classification) is the realization of the function, entities of the type MAL21 are bearers of the function

that have a function ascribed to them. According to the analysis performed in OF and illustrated in Fig. 15.4, functions are defined by means of requirements, goals and a functional item. Requirements correspond to initial conditions which must be satisfied whenever a function is realized. The goal is supposed to be achieved by the function. The functional item is a role (Loebe, 2005) played by some entity in any realization of the function. For example, in the function “to transport sugar”, illustrated in Fig. 15.4, the functional item is a sugar transporter role. The realization of a function is an entity which provides a transition from the state of the world in which the requirements of the function are fulfilled, to the state in which the goal of the function is fulfilled. This will usually be a process such as “sugar transport”, but may be any other entity. The functional item must be played in the realization of the function. The entity playing this role in the realization is the function bearer. Applied to the GO, this yields a complete picture covering all of the GO’s taxonomies, and its annotated data. The molecular function taxonomy describes the functions of gene products. These functions are realized by categories taken from the GO’s biological process taxonomy. Cellular components may participate in these processes, potentially bearing a function. However, most of the molecular functions covered by the GO are functions of the gene products that are annotated to the function category. Gene products are the bearers of the functions, and they play the role of the functional item in the realization of the function.

15.4 Biomedical Ontology Integration Through the Application of Ontological Design Principles

With the increasing number of biomedical domain ontologies there is a need for a common ontological framework in which these ontologies can be integrated. The majority of ontologies that are currently available have been developed separately, and while many adhere to the OBO guidelines this has not yet guaranteed that they are fully interoperable. There are therefore several independent efforts that attempt to integrate multiple biomedical domain ontologies.

Two different approaches are taken towards the integration of ontologies in biomedicine. The first attempts to construct upper domain ontologies based on a top-level ontology. The second constructs a core ontology with which the domain ontologies are then aligned.

Upper domain ontologies define the most general categories within a domain using the categories of a top-level ontology. For example, the category “Material structure” may be specialized to “Cell” or “Molecule”, imposing additional restrictions on these categories. A core ontology attempts to define the scope of a domain. In particular, it identifies the “core”; concepts of a domain and specifies the relation of each category or sub-domain to this “core”.

We discuss three ontologies that can be used to integrate biomedical ontologies: the BioTop (Schulz et al., 2006) ontology together with the OBO Relationship

Ontology (Smith et al., 2005), the Simple Bio Upper Ontology (SBUO),⁹ and the General Formal Ontology-Biology (GFO-Bio).¹⁰ The first two are upper domain ontologies, the latter is both an upper domain ontology and a core ontology.

15.4.1 The OBO Relationship Ontology

The OBO Relationship Ontology (OBO-RO) (Smith et al., 2005) is an ontology of the relationships that are used between entities in biomedical ontologies. Its basic ontology contains only two categories, Continuant and Occurrent. Continuants are entities which are wholly present at a single point in time, while occurrents have temporal parts and unfold through time. The OBO-RO provides a set of basic relations and gives axioms for these. Among the relations provided in the OBO-RO are the *is-a* relation, various mereotopological relations, participation, and transformation and derivation relations. For each relation, axioms specifying reflexivity, transitivity and symmetry are provided. In addition, further definitions are given in English text.

Because the OBO Relationship Ontology attempts to provide a unifying framework for all biomedical ontologies, the axioms for the relations are weak compared to more specialized theories. For example, the axioms for the part-of relationship are reflexivity, transitivity and anti-symmetry.

A number of relations are defined which are intended for use only within the biomedical domain. Among them are the relation *transformation_of* and *derives_from*. The *transformation_of* relation is a relation between two identical biological individuals at two different points in time. The *derives_from* relation relates two distinct individuals at two different points in time, and the later individual is a result of either division or fusion of the previous individual.

The OBO-RO was developed at a time when most biomedical ontologies were available as directed acyclic graphs. In these graphs, relations such as *part-of* were used as inconsistently and ambiguously. By providing these relations with consistent and unambiguous definitions, the OBO-RO aims to facilitate ontology inter-operability and to support advanced reasoning across these ontologies. New ontologies in the OBO library are required to comply with the OBO-RO.

15.4.2 BioTop and the Simple Bio Upper Ontology

The BioTop Ontology (Schulz et al., 2006) is a further development of the GENIA upper ontology. GENIA is an ontology that is intended for use in semantic annotation of texts in biological text mining (Kim et al., 2003). Several problems with GENIA's upper ontology have been identified. BioTop is an upper domain ontology for biology based on the top-level ontology BFO (Grenon, 2003), with

⁹<http://www.cs.man.ac.uk/~rector/ontologies/simple-top-bio/>

¹⁰<http://onto.eva.mpg.de/gfo-bio.html>

some concepts borrowed from DOLCE (Masolo, Borgo et al.). The relationships used in BioTop are the ones used in the OBO Relationship Ontology, plus some additional relations.

Like GENIA's upper ontology, BioTop is mainly an ontology of continuants: entities that are wholly present at each point in time at which they exist, and preserve their identity through time. Axioms are given in OWL-DL for the upper categories used in biomedical domain ontologies. For example, the category *Cell* is defined as having some Cytoplasm and no Cell as part, and having some CellularComponent and some Membrane as component.

BioTop is intended to be applied as an upper level ontology for all the ontologies listed under the OBO umbrella. By providing definitions for the upper categories of these ontologies, BioTop enforces ontological rigor and attempts to eliminate ambiguities in the use of categories. For example, when two ontologies include a Cell category, and both use BioTop for defining this Cell category, interoperability between these ontologies is made simpler.

The Simple Bio Upper Ontology (SBUO) is an upper domain ontology like BioTop. It is mainly founded in the DOLCE top-level ontology, with some ideas from BFO included. Due to the top-level ontology used, several differences distinguish the two ontologies. In particular, biological sequences like DNA sequences are abstract individuals in SBUO, while they are modeled as subclasses of molecules in BFO.

15.4.3 GFO-Bio

While BioTop and SBUO are upper domain ontologies, GFO-Bio¹¹ is both an upper domain ontology and a core ontology. This is a result of the fact that GFO-Bio attempts to make the nature of the biological domain precise, and analyzes the categories used in the upper domain ontology part with respect to their relation to biology.

GFO-Bio is based on the top-level ontology GFO (Herre, Heller et al.). The relevant features of GFO-Bio's top-level ontology that allow it to be used to analyze the nature of a domain are the inclusion of a theory of levels of reality, and explicit support for higher-order categories in GFO.

In GFO, a level of reality is a higher-order category which has as instances the categories belonging to a level (see chapter on GFO and levels). The biological level is defined using the notion of "autopoiesis" as the property of living systems (Maturana and Varela, 1991). Using the concept of autopoiesis, two principal categories are identified: Cell and Organism, which both exhibit the property of autopoietic systems on the material stratum. To the instances of these principal categories, relations taken from the top-level ontology GFO are applied to yield further categories. For example, an analysis of organisms using the subsumption relation results in a species tree. To each category in this tree, mereological relations may

¹¹<http://onto.eva.mpg.de/gfo-bio.html>

be applied to obtain a classification of anatomical parts of organisms of one species. GFO-Bio structures categories according to the relationships that must be applied to its principal categories, Organism and Cell, in order to obtain the category. This approach has been called “facet analysis” in the spirit of faceted classification used in library science.

A core ontology, such as GFO-Bio, is used differently for the integration of ontologies than an upper domain ontology. While upper domain ontologies define the upper level concepts of a domain ontology using restrictions on categories, and thereby provide definitions and restrictions for the domain categories, a core ontology specifies the relation of a domain ontology to the principal categories of the core ontology. It therefore has two main purposes: to structure sub-domains within biology according to ontological principles, and to make the nature of the biological domain precise, thereby delimiting it and allowing for a structured integration within a wider ontological framework covering multiple domains, such as chemistry. Further, it allows a faceted view of the domain of biology, starting from the principle categories of biology and exploring different facets – relationships – of these principle categories. A part of the taxonomic tree of GFO-Bio depicting the biological level and several facets or sub domains within this level is shown in Fig. 15.5. The material stratum is a higher-order category, and the biological level a sub-category of the material stratum.

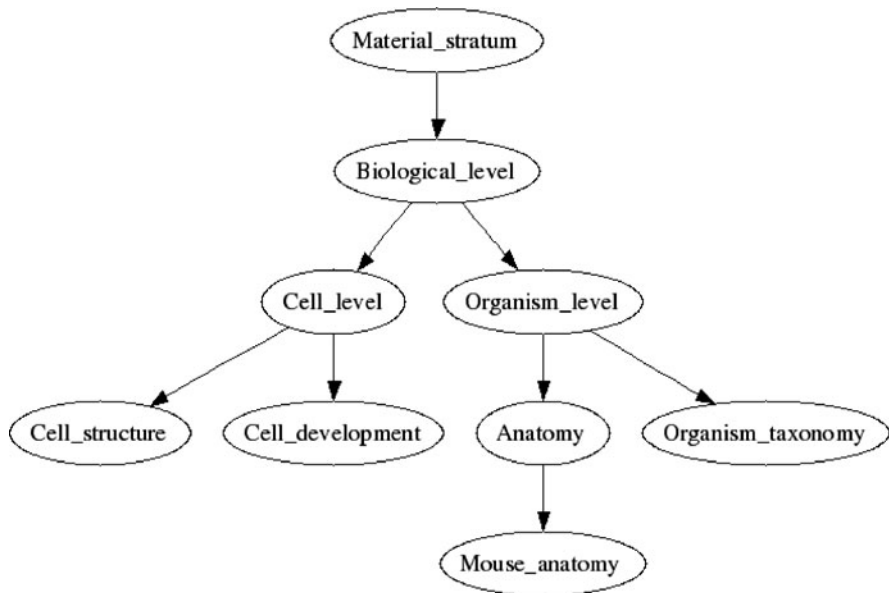


Fig. 15.5 A fragment of GFO-Bio’s classification of the biological level. The Material_stratum is considered a subcategory of GFO’s “Category”. Instances of the Material_stratum category are the categories that belong to the material stratum. Similarly, the Biological_level category has the categories and relations pertaining to the biological level as instances. The biological level is further refined to more detailed sub-levels or domains such as Anatomy

15.4.4 Defaults and Exceptions for Ontology Interoperability

Some biomedical ontologies take a particular view on the domain they cover in that they describe idealizations of the domain. Most of the anatomical ontologies fall into this class. For example the Foundational Model of Anatomy (Rosse and Mejino, 2003) describes an idealized, canonical human anatomy. A separate class of ontologies describes phenomena within a domain where these phenomena may be exceptions. An example is the Mammalian Phenotype Ontology, which is specifically designed to describe abnormal mouse phenotypes which arise from genetic modifications in mice.

Interoperability between these types of ontologies would facilitate the consistent use of biomedical data in the form of annotations, allow for queries over multiple ontologies and form a rich knowledge resource for biomedicine that could be further used in solving problems and stating hypotheses. The absence of clear principles for achieving interoperability between ontologies of this kind hinders the development of advanced applications and analysis tools based on these ontologies. As we will show in the following section, interoperability of ontologies of these different types cannot be achieved by the methods developed hitherto, and a new set of methods that transcends the framework of classical logic must be introduced in order to avoid inconsistencies and at the same time preserve the specificity of both types of knowledge.

A canonical anatomy ontology such as the Foundational Model of Anatomy contains rules such as every instance of a human body has as part an appendix. (1)

This rule does not necessarily apply to every real human body: an individual human body may *lack* an appendix as part. However, the rule describes an idealized or *canonical* human. Phenotype ontologies may describe exceptions to these idealizations. For example, an individual may both be an instance of a human body as described in the FMA (which implies an appendix as part) and an instance of the category “human body with absent appendix”. In a classical logical framework, such as those used in the OBO Relationship Ontology or in the form of the Web Ontology Language (OWL) (McGuinness, 2004), a formalization of these two statements would lead to an inconsistency. A human body in the former case has an appendix as a part, while in the latter case it does not. Instantiating both by an individual causes the inconsistency. A logical inconsistency in the formal sense can only arise when the logical functor of negation is used. This functor is hidden in concepts such as “absent X”, as used in the Mammalian Phenotype Ontology (Smith, 2005).

In order to avoid terms such as “absent X” and make the negation explicit, the **lacks** relation was introduced (Ceusters, 2007), which can be explicitly defined as:

Individual p **lacks** category C with respect to relation R if and only if there is not an x such that: xRp and x is an instance of C .

It is possible to use binary relations of the kind x **lacks- R** C instead of x **lacks** C with respect to R . For example, the fact that some individual x **lacks** a category C with respect to the relation **has-part** will be denoted as x **lacks-part** C .

The use of the **lacks** relation may cause an inconsistency when a canonical ontology and a corresponding phenotype ontology are used together with instances in a

classical logic formalism, such as first order logic or description logic. The reason is that classical formalisms enforce very strict interpretations, e.g. of quantifications like “every human”, which results in *monotonicity* of these formalisms: the inferences drawn from a classical logical theory T remain true in every extension of T with additional facts. In order to prevent inconsistencies, while at the same time preserving the intuition behind statements such as “a human has an appendix as part”, such statements in the canonical ontology must be weakened. What is required is a *nonmonotonic* logic with which the statements in a canonical ontology can be treated as true by default, but adding additional knowledge, by reference to a phenotype ontology or using a statement involving the **lacks** relation (and therefore negation), may invalidate the conclusions previously drawn.

In order to describe the nature of default relationships between two categories, new relations must be introduced, such as **CC-canonical-has-part**. For each relationship **R** between individuals, a set of relations is introduced according to Table 15.2. Then, the relationship between “human” and “appendix” becomes “human **CC-canonical-has-part** appendix”. Further, this relationship corresponds to a *default rule*:

forall $x, C1, C2$: if $C1$ **CC-canonical-has-part** $C2$ and x **IC-instance-of** $C1$ then
 by default: there exists a y : y **IC-instance-of** $C2$ and x **II-has-part** y .

Defaults rules can be formalized using answer set programs. Answer set programs are logic programs that employ two kinds of negation, strong and weak. While strong negation corresponds to classical negation, weak negation is also referred to as “default negation”. Intuitively, the weakly negated statement “not A ” means “ A cannot be proven”.

Answer set programs must be further combined with ontology representation, in order to be used within ontologies. For example, the system DLVHEX allows for a bidirectional flow of information between an answer-set program and a description logic knowledge base (Eiter, 2006). Relationships that are used in an ontology are made available to the DLVHEX system. Then, it is possible to express the necessary

Table 15.2 A schema of the relations introduced. Domain and range for the relations are encoded in the prefix of their name (e.g., **IC** means that the domain is Individual and the range Category). For each relation that is used in an imported ontology, a number of relations between categories, individuals, and between individuals and categories can be created. The **CC-canonical-R** relationship is a default relation which is accompanied by axioms in an answer set program in order to describe its semantic as a default

Relationship	Definition
x II-R y	Individuals x and y stand in the primitive relation II-R .
x IC-R y	There exists an individuals z , which is an instance of x , such that x II-R z .
x CC-R y	For all individuals a which are an instance of x : a IC-R y .
x CC-canonical-R y	For all individuals a which are an instance of x : normally, a IC-R y .
x II-lacks-R y	Not x II-R y .
x IC-lacks-R y	Not x IC-R y .
x CC-lacks-R y	For all individuals a such that: a IC-instance-of x , a IC-lacks-R y .

axioms for relations of the kind **CC-canonical-R**. For example, for the relationship **CC-canonical-has-part**, the following axiom can be added:

```
IC-has-part(X,Y) :- ind(X), class(Y), inst(X,Z),
                    CC-canonical-has-part(Z,Y),
                    not IC-lacks-has-part(X,Y), class(Z).
```

This means that if two categories Z and Y stand in the relation **CC-canonical-has-part**, and *it is not provable that X IC-lacks-has-part Y* (not `IC-lacksHasPart(X,Y)`), then it is concluded that an individual X , which is an instance of Z , stands in the relation **IC-has-part** to the category Y .

Extending biomedical ontologies with the capability for non-monotonic reasoning allows for interoperability between ontologies describing canonical knowledge within a domain and phenotype ontologies (which describe phenomena). Using a hybrid approach by combining traditional ontology representation languages such as OWL or OBO DAGs with answer set programs allows for the reuse of tools that are used in ontology development, such as Protege (Noy et al., 2003) or OBO-Edit (Day-Richter et al., 2007).

15.5 Applications

Development of the majority of the bio-ontologies has been driven by the need to order and analyze the vast amount of data collected in biological databases and acquired by experiments. The biological community has actively applied ontologies for the annotation of biological data types. A feature that distinguishes the biomedical ontologies is the vast amount of experimental data that is annotated using these ontologies. It is the combination of the ontologies with this data that has enabled large-scale biological description and discovery. A number of software packages supporting a variety of biological applications have been developed by the community, only a few of which we will discuss here. A software repository for of these packages is maintained by the Gene Ontology Consortium,¹² and while some of the tools are GO-specific, some can be used with multiple bio-ontologies.

15.5.1 Annotation and Retrieval of Data

Through formalizing the terms used for a domain and then using these for the annotation of biological data such as genes and proteins, the bio-ontologies have provided researchers with the ability to browse and retrieve data according to well known terms. In an initial approach to help researchers in genetics manage the rising number of sequences in public databases, controlled vocabularies were used

¹²<http://www.geneontology.org/GO.tools.shtml>

to assign commonly used terms to genes and proteins (eg.: The protein database Swiss-Prot (Boeckmann et al., 2003)) or cDNA libraries (Kelso et al., 2003). These controlled vocabularies were later supplemented by bio-ontologies to provide researchers with domain-specific hierarchies for the browsing and retrieval of data. For this approach no more than a simple is-a hierarchy is needed, giving a possible explanation for the simple structure of the OBO Ontologies. A standard example of a pure is-a hierarchy in biology is the classification of species, constituting an integral part in the organization of genetic information (Wheeler et al., 2003). Biological databases now make extensively use of bio-ontologies to provide controlled terms for the description of various aspects of genes and proteins.

15.5.2 Statistical Analysis of Experiments

Current technologies allow for massive parallel measurements in genetic experiments. One well-known and widely used form of experiment measures the relative amount of transcript from DNA for several thousand genes on a microarray chip (reviewed in Lockhart and Winzeler, 2000). However, the analysis and interpretation of the data generated by these experiments is often hampered by two major problems:

1. the power to draw a significant conclusion from a single measurement is low because of large technical variance in the experimental measurements, and
2. data on the level of single genes does not allow for a direct insight into the affected higher level functions of the organism.

These problems led to the development of several applications (eg.: GStat (Beissbarth and Speed, 2004) or FUNC¹³ (Pruefer et al., 2007) which make use of the simple DAG structure of the Gene Ontology in order to group genes by their annotation. This grouping increases the power to detect differences, as the measurements for multiple genes can be combined for testing. Additionally, the statistical test on the Gene Ontology DAG results in a list of significant groups. These groups are described by meaningful terms from the Gene Ontology, thus helping the user to interpret the result in terms of the biologically relevant affected processes and functions as well as the cellular localization.

While these applications vary in respect to the implemented statistical tests and the user interface, their general method is very similar. As a prerequisite, genes need to have an assigned value as the result of the experiment. These values are then collected in Gene Ontology groups according to the annotation of the gene and can be propagated to linked higher level groups in the DAG because of the True Path Rule. An appropriate statistical test is then applied to each group. Since many groups are tested for significance, the chance of a false positive result is not at the desired probability of error. This constitutes a well known problem of statistics known as multiple hypothesis testing and is addressed in several of these packages using

¹³<http://func.eva.mpg.de/>

a variety of methods for correction (Manly, 2004). The approach that we have described here is not limited to the Gene Ontology, but can be applied to any ontology that can be represented as a DAG.

15.5.3 Automatic Annotation and Community-Developed Ontologies

Given the amount of experimental and computational research required to describe gene function, the genetic bases of complex diseases, or the evolutionary history of organisms, genetics tends to be a field with a vast number of publications, often in highly focused research areas. Since the curation model used by most of the bio-ontologies requires curators to read literature in order to extract the ontological terms and annotations, this leads to a bottleneck in the curation of these ontologies. Generally the curators read a defined subset of publications to create annotations. Two alternative approaches to addressing this challenge have been undertaken.

15.5.3.1 Automatic Annotation

Using methods from computer linguistics and information extraction several authors (Hirschman, 2005) have explored automating the search for relevant publications for each term in an ontology. Information extraction from biological texts is a powerful means to increase the coverage of ontologies and their annotations. Such approaches may also have the ability to verify their correctness, providing increased confidence in the automatically generated results. Several sophisticated software implementations have been developed to extract information about e.g. gene and protein functions from biomedical literature (Camon, 2005). However, while information extraction from biomedical texts can quickly provide huge amounts of structured information that potentially can be added to ontologies as categories, relationships or annotations, manual verification and quality assurance based on human input is always beneficial.

15.5.3.2 Community Development

A very recent development to increase the amount of captured information from publications is the use of Wikis (Leuf and Cunningham, 2001) which aim to involve the community directly in the curation process. While no fully fledged Wiki for this purpose exists currently, there are several proposed methods, spanning several degrees of formalism for the captured information. A natural way of applying the current Wiki technology is to allow natural language descriptions for each gene, to supplement the genome databases with further information gained from experts (Wang, 2006). Such a wiki does not yet exist, but there are proposals to provide such functionality via a new project called WikiProteins (Giles, 2007).

The most formal approach to date is currently under development by Hoehndorf et al. (Hoehndorf et al., 2006). Within this wiki users are able to edit annotations and add or modify concepts in the ontology. Additional to the natural-language

aspect, the wiki provides a way to add formal n-ary relations with subject, object and additional mandatory and obligatory roles.

A background Core Ontology (GFO-BIO) together with a reasoner are used to ensure that the information in the wiki is reasonably accurate. The formal entries must be typed on the basis of the Core Ontology and the reasoner is applied to limit the entries to those that are consistent with already entered information.

15.5.4 Reasoning for Experimental Hypothesis Testing

There are few advanced applications of the bio-ontologies, perhaps because many still lack the required formality to support such applications. A recent and interesting example of the use of bio-ontologies in the formulation and testing of experimental hypotheses is the Robot Scientist project (Soldatova et al., 2006). The Robot Scientist is a robotic laboratory system able to design, perform and evaluate biological experiments in a microbiological laboratory. Based on a general ontology of experiments, EXPO, (Soldatova and King, 2006) in which data and metadata about all aspects of the experiment are captured, the robot is iteratively able to formulate hypotheses, physically carry out the experiments, and then evaluate results in order to use the information gathered in the next experiment.

A second example is Hybrow (Racunas et al., 2004). HyBrow is a system to design and evaluate hypotheses and verify their consistency with available biomedical knowledge. It uses an event-based ontology for representing biological processes in the background. A prototypical implementation is available.¹⁴

15.6 Summary and Conclusions

The research field of bio-ontologies has grown rapidly in the past 10 years. This is a direct result of the need in the biomedical research community to define and share the vocabulary used for the description of the growing quantities of biological data being generated. With increasing amount of data more difficulties were encountered in managing, sharing and integrating these data. While several early projects, notably BioCyc (Karp et al., 2005) and GALEN (Rector and Nowlan, 1994), provided ontologies for parts of the biomedical domain, the newer, “light-weight” ontologies such as the Gene Ontology were developed by biologists to solve the specific problems that they face in daily research activities. These ontologies were therefore designed to address a specific, restricted set of problems – mainly annotation and database integration – and initially tended to sacrifice formal logical and ontological rigor to achieve this goal in a reasonable time-frame. Over time, and following the success of ontologies like the Gene Ontology, biomedical

¹⁴www.hybrow.org/

ontologies are being gradually extended, formal foundations laid, and ontological principles applied. This is being done in an effort to facilitate interoperability between the various ontologies that were developed for distinct, but related domains. Ultimately, these improvements will enable the automatic detection and prevention of inconsistencies, and automatic extraction of implicit knowledge. The development and application of top-level ontologies, the construction of upper-domain and core ontologies, and the unification of the relationships used in the various biomedical sub-domains are all significant steps in the construction of a unified biomedical knowledge base. As an increasing amount of knowledge is formalized, the application of ontologies and other biomedical knowledge bases for the generation of biological and biomedical hypotheses, their verification, the automatic planning and evaluation of experiments and the detection of conflicting biomedical claims may become possible. The community-wide adoption of software implementations that use ontologies for the statistical analysis of experimentally generated gene lists (Beissbarth and Speed, 2004; Pruffer et al., 2007), or the identification of protein functions using ontologies (Wolstencroft, 2006) indicate that ontologically-based applications are a welcome addition to the biologists data generation and analysis toolset. Biomedicine is likely to remain a largely data-driven discipline that capitalizes on the intuition, experience and intellect of the biomedical researcher. However, parts of the field are amenable to becoming knowledge-driven disciplines. It therefore seems likely that ontology-based biomedical knowledge-bases will play an increasingly important role in modern biomedicine, and act a motivating force in computational logics, Semantic Web technologies, and for foundational ontological research.

References

- Ashburner, M., et al. 2000. Gene ontology: Tool for the unification of biology. The gene ontology consortium. *Nature Genetics* 25(1):25–29.
- Bada, M., and L. Hunter. 2007. Enrichment of OBO ontologies. *Journal of Biomedical Informatics* 40(3):300–315.
- Beissbarth, T., and T.P. Speed. 2004. Gostat: Find statistically overrepresented gene ontologies within a group of genes. *Bioinformatics* 20(9):1464–1465.
- Bodenreider, O., et al. 2003. Evaluation of WordNet as a source of lay knowledge for molecular biology and genetic diseases: A feasibility study. *Studies in Health Technology and Informatics* 95:379–384.
- Boeckmann, B., et al. 2003. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research* 31(1):365–370.
- Burek, P., et al. 2006. A top-level ontology of functions and its application in the open biomedical ontologies. *Bioinformatics* 22(14):e66–e73.
- Camon, E., et al. 2004. The gene ontology annotation (GOA) database: Sharing knowledge in uniprot with gene ontology. *Nucleic Acids Research* 32(Database issue):D262–D266.
- Camon, E.B., et al. 2005. An evaluation of GO annotation retrieval for BioCreAtIvE and GOA. *BMC Bioinformatics* 6(Suppl 1):17.
- Ceusters, W., et al. 2007. Negative findings in electronic health records and biomedical ontologies: A realist approach. *International Journal of Medical Informatics* 76(Suppl 3):S326–S333.

- Day-Richter, J., et al. 2007. OBO-edit – An ontology editor for biologists. *Bioinformatics* 23(16):2198–2200.
- Eiter, T., et al. 2006. Dlvhex: A system for integrating multiple semantics in an answer-set programming framework. Proceedings 20th Workshop on Logic Programming and Constraint Systems (WLP 06).
- Field, D., et al. 2006. Meeting report: eGenomics: Cataloguing our complete genome collection II. *OMICS: A Journal of Integrative Biology* 10(2):100–104.
- Giles, J. 2007. Key biology databases go wiki. *Nature* 445(7129): 691.
- Golbreich, C., and I. Horrocks. 2007. The OBO to OWL mapping, go to OWL 1.1! Proceedings of OWL-ED 2007.
- Grenon, P. 2003. BFO in a nutshell: A bi-categorical axiomatization of BFO and comparison with DOLCE.
- Grenon, P., et al. 2004. Biodynamic ontology: Applying BFO in the biomedical domain. *Studies in Health Technology and Informatics* 102:20–38.
- Harris, M.A., et al. 2004. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research* 32(Database issue):D258–D261.
- Herre, H., et al. General formal ontology (GFO): A foundational ontology integrating objects and processes. Part I: Basic principles.
- Hirschman, L., et al. 2005. Overview of BioCreAtIvE: Critical assessment of information extraction for biology. *BMC Bioinformatics* 6(Suppl 1):S1.
- Hoehndorf, R., et al. 2006. A proposal for a gene functions Wiki. On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, 669–678.
- Karp, P.D., et al. 2005. Expansion of the bioCyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Research* 33(19): 6083–6089.
- Kelso, J., et al. 2003. eVOC: A controlled vocabulary for unifying gene expression data. *Genome Research* 13(6A):1222–1230.
- Kim, J.D., et al. 2003. GENIA corpus-semantically annotated corpus for bio-textmining. *Bioinformatics* 19 Suppl 1:i180–i182.
- Kumar, A., and B. Smith. 2004. Enhancing GO for the sake of clinical bioinformatics. Proceedings of Bio-Ontologies Workshop.
- Kumar, A., et al. 2003. The unified medical language system and the gene ontology: Some Critical Reflections. *KI2003: Advances in AI*: 135–148.
- Leuf, B., and W. Cunningham. 2001. *The wiki way: Quick collaboration on the web*. Boston, MA: Addison-Wesley.
- Lewis, S. E. 2005. Gene ontology: Looking backwards and forwards. *Genome Biology* 6(1):103.
- Lockhart, D.J., and E.A. Winzeler. 2000. Genomics, gene expression and DNA arrays. *Nature* 405(6788):827–36.
- Loebe, F. 2005. Abstract vs. social roles: A refined top-level ontological analysis. Proceedings of the 2005 AAAI fall symposium roles, An Interdisciplinary Perspective: Ontologies, Languages, and Multiagent Systems', AAAI.
- Manly, K.F., D. Nettleton, et al. 2004. Genomics, prior probability, and statistical tests of multiple hypotheses. *Genome Research* 14(6):997–1001.
- Masolo, C., et al. 2003. Wonderweb deliverable D17. The WonderWeb Library of Foundational Ontologies and the DOLCE ontology.
- Maturana, H.R., and F.J. Varela. 1991. *Autopoiesis and cognition: Realization of the living (Boston studies in the philosophy of science)*. Berlin: Springer.
- McGuinness, D.L., and V.H., Frank. 2004. OWL web ontology language overview.
- Noy, N.F., et al. 2003. Protege-2000: An open-source ontology-development and knowledge-acquisition environment. *AMIA Annual Symposium Proceedings*: 953.
- Pruefer, K., et al. 2007. FUNC: A package for detecting significant associations between gene sets and ontological annotations. *BMC Bioinformatics* 8:41.
- Pruefer, K., et al. 2007. FUNC: A package for detecting significant associations between gene sets and ontological annotations. *BMC Bioinformatics* 8:41.

- Racunas, S.A., et al. 2004. HyBrow: A prototype system for computer-aided hypothesis evaluation. *Bioinformatics* 20(Suppl 1):i257–i264.
- Rector, A.L., and W.A. Nowlan. 1994. The GALEN project. *Computer Methods and Programs in Biomedicine* 45(1–2):75–78.
- Rosse, C., and J.L. Mejino, Jr. 2003. A reference ontology for biomedical informatics: The foundational model of anatomy. *Journal of biomedical informatics* 36(6):478–500.
- Schulz, S., et al. 2006. Towards an upper level ontology for molecular biology. *AMIA Annual Symposium Proceedings*: 694–698.
- Schulze-Kremer, S. 1998. Ontologies for molecular biology. *Pacific Symposium on Biocomputing* 3:695.
- Schulze-Kremer, S. 2002. Ontologies for molecular biology and bioinformatics. *In Silico Biology* 2(3):179–193.
- Smith, B., et al. 2003. The ontology of the gene ontology.
- Smith, B., et al. 2004. On the application of formal principles to life science data: A case study in the gene ontology. Proceedings of DILS 2004 (Data Integration in the Life Sciences), Springer.
- Smith, B., et al. 2005. Relations in biomedical ontologies. *Genome Biology* 6(5):R46.
- Smith, C.L., et al. 2005. The mammalian phenotype ontology as a tool for annotating, analyzing and comparing phenotypic information. *Genome Biology* 6(1):R7.
- Soldatova, L.N., and R.D. King. 2006. An ontology of scientific experiments. *Journal of the Royal Society Interface* 3(11):795–803.
- Soldatova, L.N., et al. 2006. An ontology for a robot scientist. *Bioinformatics* 22(14):e464–e471.
- Wang, K. 2006. Gene-function wiki would let biologists pool worldwide resources. *Nature* 439(7076):534.
- Wheeler, D.L., et al. 2003. Database resources of the national center for biotechnology. *Nucleic Acids Research* 31(1):28.
- Whetzel, P.L., et al. 2006. The MGED ontology: A resource for semantics-based description of microarray experiments. *Bioinformatics* 22(7):866–873.
- Wolstencroft, K., et al. 2006. Protein classification using ontology classification. *Bioinformatics* 22(14):e530–e538.

Chapter 16

The Ontology of Medical Terminological Systems: Towards the Next Generation of Medical Ontologies

Heinrich Herre

16.1 Introduction

Integrating, processing and applying the rapidly expanding information generated in medicine, bio-medicine and biology is one of most challenging problems facing research in these fields today. As the volumes of experimental data and knowledge increases, there is a growing need for supporting formal analyses of these data and pre-processing knowledge for further use in solving problems and stating hypotheses. To satisfy this pressing need we must increase the formality, expressivity and transparency of medical terminologies and biomedical ontologies. Achieving these goals requires the precise and formal characterization of medical, biological and biomedical data and knowledge, as well as their correct representation in computational form.

T Gruber introduced the term *ontology* to computer science (Gruber, 1993). He defined ontology as a formal specification of a conceptualization. Computer scientists initially promoted ontologies as a technology for overcoming problems of heterogeneity. Now, biologists and medical computer scientists employ ontologies to consistently represent and process biological and medical information and knowledge. Ontologies have proven useful and beneficial for integrating and sharing data, managing terminology, reusing knowledge and supporting decisions. Bodenreider, et al. observed that as the field progresses, ontology's philosophical underpinnings play a growing role, though they do not address these foundational considerations (Bodenreider, 2006). Given that the number of ontologies is rapidly growing, there is an urgent need to create and establish scientific techniques and methods, conceptual tools, and computer-based supporting systems for building ontologies and to establish objective metrics and criteria for measuring and evaluating their quality.

H. Herre (✉)

Research Group Onto-Med, IMISE, University Leipzig, Leipzig, Germany
e-mail: heinrich.herre@imise.uni-leipzig.de

Contribution for the TAO-Volume, (Theory and Application of Ontology)

What follows is a critical analysis of the current situation, particularly with regard to the significance of logic, artificial intelligence and philosophy for ontology research, in an attempt to close the gap that remained open since (Bodenreider, 2006). A broader framework is needed to analyze and represent the relevant phenomena occurring in the field of terminologies and ontologies. Such a framework should be based on logic, artificial intelligence, linguistics and philosophy. Logic contributes to a rigorous formalization of biomedical content, linguistics plays a role in the analysis of natural languages texts, artificial intelligence is relevant for knowledge representation, inference procedures and integration methods, and, finally, philosophical ontology provides a framework for the categorization of the world. The research group Onto-Med at the IMISE, University of Leipzig, is constructing the architecture for and developing the constituents of such a framework. Finally, future avenues of research will be explored, particularly, possible features of the next generation of medical ontologies.

First, in Section 16.2, the notion of an ontology and a terminological system is outlined. Furthermore, to make the paper self-contained, some basic notions of GFO (General Formal Ontology) are summarized which are used in the subsequent sections. Section 16.3 expounds some basic ideas on domains and conceptualization which are needed in the following sections. Section 16.4 applies the ideas of Section 16.3 to some recent results on medical terminological systems. Section 16.5 provides an overview about some important medical terminological systems and includes a preliminary ontological analysis of these systems. Section 16.6 concludes with recommendations for future research, and expounds ideas which might be relevant for developing the next generation of medical ontologies.

16.2 Terminological Systems and Ontologies

Here, an ontology should be understood as an information or knowledge system comprised of terminological systems as well as formal theories in a logical language. The similar term *formal ontology* should be considered as a science that addresses the systematic development of axiomatic theories describing forms, modes and views of being at different levels of abstraction and granularity. The science of formal ontology combines the methods of mathematical logic with analyses and principles of philosophy, but also with the methods and principles of other sciences, in particular artificial intelligence, cognitive psychology and linguistics.

Ontologies, like information systems, are roughly classified into terminological systems, frame-based systems and logical theories. Basic constituents of an ontology are categories, relations, objects, and symbolic structures that serve as designations. On the formal syntactic side an ontology is a symbolic system composed of strings and words. These strings denote categories, relations or objects. Categories are entities that are expressed by predicative terms of a formal or natural language and that can be predicated of other entities. The most important types

of categories are concepts, which are basic constituents of terminological systems. Concepts, designations and objects form the so-called *semiotic triangle* (Campbell et al., 1998).

Terminological systems, in general, are based on concepts; they relate concepts of a particular domain to one another, and provide their terms and possibly their definitions and codes. Concepts are cognitive constructs that are sometimes intuitively called *units of thought*; hence, concepts belong to the mental-psychological stratum of the world,¹ according to the GFO-framework presented in (Herre et al., 2006) and Poli's theory of levels of reality (Poli, 2001). *Linguistic labels*, called terms, are used to designate concepts. *Codes* are constructed of letters, numerals or combinations thereof; they can be used to designate concepts in a computerized system. Both, linguistic labels and codes, are symbolic structures. Usually, linguistic labels are taken from a natural language; hence, they possess, aside from the designation function, an intension that is relevant for the comprehension of a term or word. The semiotic triangle is useful for depicting interrelations between term, thought (mental state, epistemic state, and meaning) and an object (as referent). The relation between a term or word and an object is indirect. A subject (a person, an agent) processes a word, which evokes a corresponding thought (cognitive state, epistemic state) and then links this thought to an object. We emphasize that a subject can understand and grasp a subject-independent object only through a mental state, a unit of thought.

The notion of terminology uses partly the exposition of N.F. de Keizer in (2000a), and (de Keizer et al., 2000b). A *terminology* is a list of terms referring to concepts in a particular domain. In a simple terminology, no ordering is assumed. A *thesaurus* is a terminology in which terms are ordered, alphabetically or systematically. Here, concepts can possibly be described by more than one (synonymous) term. When definitions for the concepts in a terminology are added, it becomes a *vocabulary* or *glossary*. A *nomenclature* is a system of terms, composed according to pre-established composition rules for constructing new complex concepts. A *taxonomy* is an arrangement of categories according to the "is-a" relationship from a subordinate category to a super-ordinate category.

Logical theories are based on a model-theoretical language. A model-theoretical language consists of a structured vocabulary $V(O)$, called an *ontological signature*, and a set of axioms $Ax(O)$ in which $V(O)$ is formulated in a formal language $L(O)$. Hence, an ontology (understood as a formal object) is a system $O = (L, V, Ax)$; the symbols of V denote categories and relations between categories or between their instances. L represents an operator that associates a set of expressions $L(V)$, which are usually declarative formulas, to a vocabulary V . Assuming the following conditions, $V \subseteq V'$ implies $L(V) \subseteq L(V')$, and $L(L(V)) = L(V)$. An ontology

¹The notion "unit of thought" is very vague. This and similar notions, such as "epistemic state", "perception", "mental state", "cognitive state" and "meaning" belong to the mental-psychological stratum, whose investigation is a subject for future research (Albertazzi, 2001; Albertazzi, 2003; Poli, 2006).

may be augmented by a derivability relation, denoted by \vdash . Such an ontology then becomes a knowledge system $(L, V, Ax, Mod, \vdash, \models)$. We use the standard notions of model theory and logic as set forth in (Chang and Keisler, 1977): $Mod(X)$ denotes the class of models of a theory X , \models is the semantical consequence relation, and \vdash denotes an inference relation. Throughout the paper, an ontology is considered as a system (Cat, Rel, Ax) consisting of a set of categories (Cat), a set of relations (Rel), and a set of axioms (Ax) that specify connections between the categories and describe properties of the relations in Rel. The simplest form of an axiom is presented as a relational link between categories. The precise meanings of these links must be explained using formulas from a language. Furthermore, among the categories there are individual concepts that represent objects.

In the remaining part of this section we summarize basic features of GFO (General Formal Ontology) which are used throughout this paper. GFO is a foundational ontology that is under development by the research group Onto-Med (Ontologies in Medicine) at the University of Leipzig. A detailed exposition of GFO is set forth in (Herre et al., 2006), and an outline is presented in (Herre, 2010). *Entity* is defined here as everything that exists, where existence is understood in the broadest sense. Categories are entities expressed by predicative terms of a formal or natural language that can be predicated of other entities. Predicative terms are linguistic expressions T that state the conditions $Cond(T)$ to be satisfied by an entity. Entities are primarily distinguished as sets and items. Items are classified into categories and individuals. Categories are classified into *concepts*, *universals*, and *symbol structures*. Categories can be conceived without a forced commitment to realism, conceptualism or nominalism; this is justified by a new kind of realism, called *integrative realism* which is expounded in (Herre, 2010). Individuals are entities that are not instantiable; they are divided into pure space-time entities and non-space-time individuals. Non-space-time individuals are classified into concrete and abstract individuals. Concrete individuals exist in time or space, whereas abstract individuals do not.

With regard to the relationship between individuals, time and space, there is a well-known philosophical distinction between endurants and perdurants. Unlike the vague notion of an endurant, GFO uses, instead of an endurant, the more precise notion of a presential. A presential captures one aspect of an endurant: it is an individual that is entirely present at a particular time-point. GFO accounts for persistence using a suitable universal whose instances are presentials. Such universals are called persistants. These do not change and can be used to explain how presentials with different properties at different times can be the same. Therefore, persistants are special categories that can be instantiated. We claim that for every persistant P of a certain subclass of persistants, there exists an individual q , called a *perpetuant*, which persists through time and is related to the time-points of its duration by a relation $exhib(q, a, t)$. The relation $exhib(q, a, t)$ means that (the perpetuant) q exhibits the presential a at a time-point t . Perpetuants are cognitive constructions of the mind whose existence can be justified by Gestalt theory (Wertheimer, 1912).

16.3 Domains and Graduated Conceptualizations

This section outlines a preliminary account of the structure of domains and their conceptualizations. In particular, we introduce the notion of a *graduated conceptualization*. The formation and emergence of domains is a result of the evolution of scientific knowledge, but also the product of common sense reasoning, social awareness and both, philosophical contemplations and reflections.

A domain D , is determined by a set of objects $\text{Obj}(D)$, by a set V of views at $\text{Obj}(D)$, and by a set CP of classification principles for $\text{Obj}(D)$. Hence, a domain is represented as a three component system, $D = (\text{Obj}(D), V, CP)$. The notion of view is used in an informal, intuitive sense, whereas the classification principle can usually be made more precise. In understanding, acquiring and representing the knowledge about a domain, we use categories and relations between them, and must specify the domain's individuals and their fine-structure. Hence, two additional constituents are associated to a domain: a set of categories of D , denoted by $\text{Cat}(D)$ and, a set of relations of D , denoted by $\text{Rel}(D)$. These additional constituents are influenced by the views of $\text{Ind}(D)$ and the classification principles of D . The system $\text{Concept}(D) = (\text{Ind}(D), \text{Cat}(D), \text{Rel}(D))$ can be conceived as a detailed form of a conceptualization of the domain D in the sense of (Gruber, 1993). This approach to conceptualizations supports the ideas McCray expounded (McCray, 2006), assuming that the categorical system $\text{Cat}(D)$, the relations in $\text{Rel}(D)$, and also the classification principles $CP(D)$, depend on the more deeply rooted world view of its designer, including the purpose for which the categorical system is generated. The relations in $\text{Rel}(D)$ can be classified into relations between individuals, called individual relations, and relations between categories, called categorical relations, and finally as mixed relations which have as elements both individuals and categories.² An ontology of a domain is based on a conceptualization; it is determined by adding axioms describing inter-relations between the categories and properties of relations. An ontology Ont can be presented as a system $\text{Ont} = (\text{Concept}(D), \text{Ax}(\text{Concept}(D)))$, where $\text{Ax}(\text{Concept}(D))$ denotes the set of axioms about the conceptualization $\text{Concept}(D)$. The simplest axioms are represented by relational links between categories.

Let D be a domain and $\text{Cat}(D)$ be the set of categories associated to D . The conceptualization of D , and hence the system $\text{Cat}(D)$, is not uniquely determined and may change over time.³ This phenomenon is discussed in (McCray, 2006), in which a number of examples is collected and analyzed. Since a domain depends on a view,

²A more fine-grained system of types for relations may be introduced.

³Recently, a principle of *coordinated evolution* is discussed in (Smith et al., 2007). One of the included rules, called orthogonality, is stating that for every domain there should be only one ontology. Such a principle must be rejected, it contradicts elementary evolution principles in science. There is great diversity of ontologies pertaining to the same domain which are determined by the conceptualization, by the axioms selected, and by the expressivity of the language in which these axioms are formulated.

a set of individuals and a classification principle, there are also cases for which the domain changes with respect to the view and the classification principle, but leaves the objects fixed. Examples for these kinds of changes and deviations may be found in the biological and medical domain. The entire biological domain changes permanently by creating new sub-domains, introducing new concepts and by modifying the content of existing concepts. Recent developments concern the cladistic taxonomy of biology, which differs from classical systems in biology which group species and other biological categories according to shared features and characteristics (Hennig, 1950; Hennig, 1975; Wiley, 1981). The cladistic taxonomy is based on another classification principle, arranging the taxa in an evolutionary tree. A new nomenclature, the PhyloCode, currently under development, is intended to address the evolutionary tree.

The categories of $\text{Cat}(D)$ are divided into a set of principal categories of D , denoted by $\text{PrincCat}(D)$ of D , into a set of elementary categories of D , designated by $\text{ElemCat}(D)$, into a set of aspectual categories of D , symbolized by $\text{AspCat}(D)$, and into a linguistically defined category, denoted by $\text{LingCat}(D)$. These sets of categories form an increasing chain, i.e. suppose that $\text{PrincCat}(D) \subseteq \text{ElemCat}(D) \subseteq \text{AspCat}(D) \subseteq \text{LingCat}(D)$. The system $(\text{PrincCat}(D), \text{ElemCat}(D), \text{AspCat}(D))$ is called a *graduated conceptualization* for the domain D . The principal categories are the most fundamental of a domain. For the biological domain, the organism category is accepted as principal. Identifying a domain's principal categories is usually the result of an evolution of domain-knowledge.

The elementary categories of a domain are introduced and determined by a classification based on the domain's classification principles; they usually present a taxonomy. There is a great variety of combinations of the classification principles being applied to specify elementary categories. A domain D is called simple if it has only one view and one classification principle, and if the taxonomy based on $(\text{Obj}, V, \text{CP})$ exhibits a tree-like structure. If the domain has multiple views, then the taxonomic ordering of elementary concepts cannot be assumed to be tree-like. Multiple views can explain the occurrence of multiple inheritances.

In addition to the elementary categories, there is an open-ended set of aspectual categories derived from the fact that any entity stands in many relations to other entities. Aspectual categories are derived from elementary categories by aspectual composition and deployment. Presently, there are no clearly formulated and complete principles for defining and deriving aspectual categories. The notion of aspectual analysis corresponds roughly to the notion of facet analysis in (Ranganathan, 1962), whereas the notions of aspectual composition and deployment are concerned with the construction of new categories from particular constituents.

New concepts can be introduced along dimensions or basic aspects. Basic aspects are categories or basic relations of a top-level ontology, which is in the sequel GFO. An intuitive, informal relation aspect (X, Y, Z) means: X is a domain category, Y a basic category or a basic relation of GFO and Z a category derived from X using the category or relation Y in the role of an aspect. Therefore, Z is an aspectual category of X via Y .

16.4 Analyses of Terminological Systems

An ontological analysis of a medical terminological system should uncover some of its basic features. These features include the presupposed classification principles, the sub-domains and core categories, the distinction between elementary and aspectual concepts and an explication of the used relations. This analysis will provide a necessary basis for constructing knowledge extensions of terminological systems and biomedical ontologies, which will enhance their applicability to the life sciences.

The fragmentary exposition in Section 16.3 of the architecture of a conceptualization can be used to clarify and analyze some problems recently raised in the literature. In the paper (Bodenreider et al., 2004a) the authors analyze and investigate phenomena that they call the intrusion of epistemology into biomedical terminology. They find that every term in terminological systems used in clinical practice or in biomedical research is assumed to designate a corresponding class or concept. They then claim that only some of these terms represent classes (universals), while many others are merely assertions about such classes that are formulated in order to meet current practical coding requirements. They contend that genuine classes must reflect a special categorization principle based on the notion of similarity with respect to a set of qualities that do not depend on relations. However, there are categories that are not based on such a principle. A domain's categories are determined by the associated classification principles, and the exorbitant diversity of domains excludes a universal invariant basis for categorization.⁴

Bodenreider, et al.'s analyses (2004a) are based on an overly restricted understanding of categories and their formation, which hampers a comprehensive and well-established analysis of the observed phenomena. The concepts described in (Bodenreider et al., 2004a) as illustrating example are actually aspectual derivatives of elementary concepts. These aspectual derivatives are important for presenting knowledge about a domain. Aspectual categories include epistemologically loaded concepts as a special case. Furthermore, from an ontological point of view, epistemology is included in the mental-psychological stratum of reality.⁵ Four examples are evaluated below.

Example 1(Variation).

(Bodenreider et al., 2004a): The terms *Gram-negative bacteria* and *Gram-positive bacteria* do not correspond to particular classes in reality; hence, they do not present classes.

⁴E. Rosch doubts whether an invariant basis exists for categorization; hence, there may be a flaw in the so-called classical theory, according to which the invariant features are categorized (Cohen and Lefebere, 2005; Rosch, 1975).

⁵Ontology questions what an entity *is* and what *mode of existence* it exhibits. Epistemology questions *how* a subject relates to an object and *how* knowledge is acquired and processed. But ontology may raise questions of existence about epistemology. In turn, epistemology may ask questions regarding how an ontologist, as a subject, relates to the reality. This process may be iterated on both sides. For purposes of this chapter, the ontological view to which epistemology belongs is the mental-psychological stratum.

GFO-analysis The notions of Gram-negative bacteria and Gram-positive bacteria are aspectual concepts (extensionally) subsumed by the concept bacteria. They are not introduced by the classification principles applied to bacteria, but are aspectual derivatives of the concept bacterium. Aspectual derivatives can be important for further sub-classification of a category C. For example, it might be important to subdivide a certain group of individuals into subgroups according to certain properties and relational conditions. A specialist for environmental studies might be interested in investigating elephants living in a certain location of Africa during a particular time interval. This is a concept, but it is not derived from a biological classification principle.

Example 2(Conjunction).

(Bodenreider et al., 2004a): In ICD-9-CM one may find the following term:

Tuberculosis of adrenal glands, tubercle bacilli not found (in sputum) by microscopy, but found by bacterial culture. This expression does not present a class, because it includes epistemological information. Only the term *tuberculosis of adrenal glands* corresponds to a class.

GFO-analysis: The cited expression denotes a concept that is an aspectual derivative of the elementary concept *Tuberculosis of adrenal gland*. Similarly, the following terms can be analyzed: *Closed skull fracture without intracranial injury*, *Open skull fracture without intracranial injury*, *Closed skull fracture with intracranial injury* and *Open skull fracture with intracranial injury*. These aspectual concepts may be adequately represented in the GFO framework.

Example 3(Modality).

(Bodenreider et al., 2004a): *Definite tubo-ovarian abscess, Probable tubo-ovarian abscess, Possible tubo-ovarian abscess.* The authors observe that this is a completely different issue; these terms do not describe subclasses of *tubo-ovarian abscess*.

GFO-analysis: These terms do not classify types of tubo-ovarian abscess, but they are related to *another domain*, the domain of *diagnostic decisions, estimation, and evaluations*. Any disease described in ICD-10 could be, in principle, augmented with an additional but separate set of concepts pertaining to diagnostic decisions, estimations and evaluations. These entities span a new domain that exhibits inter-relations between the domain of diseases themselves.⁶

Example 4(Vagueness and underspecification)

(Bodenreider et al., 2004a): *Open fracture of unspecified cervical vertebra, Concussion with loss of consciousness of unspecified duration, Replacement of unspecified heart valve, Poisoning by unspecified drug or medicinal substance.* These terms do not describe classes.

⁶It might be possible to explicate and represent this additional information adequately, but for current ICD-10 use this would generate a superfluous overhead. The situation changes, of course, if knowledge extensions of ICD are established, which can be used for inferences. In this case a clear separation of these distinct domains and an explication of their inter-relations are necessary.

GFO-analysis. These concepts are associated to the domain of diagnostic decisions and evaluations and can be treated as Example 3.

16.5 Medical Terminological Systems

This section provides a brief overview of several important medical terminological systems and discusses some of their ontological features.

16.5.1 ICD

The *International Statistical Classification of Diseases (ICD)* is one of the most important international medical terminological systems; it was first issued in 1893. Its sixth revision was in 1948, and since this time it has been maintained by the World Health Organization (WHO). The current version is the tenth revision (ICD-10), which was issued in 1992. The initial aim of the ICD was to provide an international classification of death causes in order to produce internationally uniform and thus comparable mortality statistics. The WHO family of international classifications also includes other systems, notably the ICF (International Classification of Functioning, Disabilities and Health) and ICHI (International Classification of Health Interventions).

The 22 main sub-categories of ICD-10 include, among others, diseases of the blood and blood-forming organs (D50–D89), endocrine, nutritional and metabolic diseases (E00–E90), mental and behavioral disorders (F00–F99), diseases of the nervous system (G00–G99) and certain infections and parasitic diseases (A00–B99). We present some preliminary observations about ICD-10 and consider the sub-domains I–XVII (codes A00–Q99). A core ontology of ICD-10 must explicate what sub-domains I–XVII address. Six of these domains are classified with respect to systems (nervous system, circulatory system, respiratory system, digestive system, musculo-skeletal system, genito-urinary system), three pertain to special organs (eye, ear, skin), one domain relates to infectious diseases (A00–B99) and one domain addresses mental and behavioral disorders (F00–F99). Sub-domain level categories Level(*i*), *i* = I, . . ., XVII, may be introduced; their instances are subsumed by the corresponding chapters. The instances of a level category Level(*i*) in ICD-10 exhibit a taxonomic structure.

Consider the domain of infections and parasitic diseases (A00–B99) and the associated domain-level category level(I). One of the classification principles is based on the pathogens that cause the disease. Hence, the concepts in Level(I) have a taxonomic concept, “infectious and parasitic diseases” (diseases caused by pathogens). With respect to the classification of pathogens, elementary concepts and aspectual derivatives can be explained. Among the elementary concepts are *viral diseases*, *bacterial diseases*, *rickettsioses*, *mycoses* and *protozoal diseases*. These concepts are further classified with respect to a sub-classification of pathogens. A number of concepts in Level(I) can be easily identified as aspectual derivatives of

elementary concepts, among them *intestinal infectious diseases* (aspect of localization), *infections with a sexual mode of transmission* (aspect of transmission) and *viral infections characterized by skin and mucous membrane lesions* (aspect of location and symptoms).

These hierarchies are associated with different domains, whose ontological analysis is a challenging task. According to the principles expounded in Section 16.3, sub-domains and their corresponding classification principles must be identified. These exhibit the basis for a division into elementary and aspectual concepts.

16.5.2 SNOMED-CT

SNOMED (*Systematized Nomenclature of Medicine*) is a graph-based system whose basic components are concepts and relationships, together with a description logic formalism for defining complex concepts. “SNOMED Clinical Terms” (SNOMED CT) is the latest in a long series of works of terminology developed and distributed by the College of American Pathologists (CAP) for the purpose of encoding, storing and retrieving information on disease and health. Beginning in the mid-1990s, the CAP started a radical re-engineering of SNOMED with the understanding that manual coding would become an activity of the past, and that substantial changes were required to support increasingly sophisticated electronic systems in healthcare and public health. This work enabled CAP to publish the SNOMED Reference Terminology (RT) in 2000. An even larger transformation occurred when SNOMED RT was merged with the UK National Health Service’s (NHS) Clinical Terms version (CTV3), resulting in the first release of SNOMED Clinical Terms (CT) in January 2002. Since that time there have been an additional four releases, one every 6 months. In 2003, the US Government licensed SNOMED CT, and the National Committee on Vital and Health Statistics (NCVHS) recommended its use as the general terminology for patient medical record information in the US. In the UK, SNOMED CT is a draft national standard and a key element of the NHS National Program for IT. Thus, SNOMED is being developed with serious expectations and demands for practical use.

SNOMED consists of eighteen independent hierarchies reflecting, in part, the organization into axes such as Disease, Drugs, Living organism, Procedure and Topography. Among these top-level concepts belong *clinical finding*, *procedure*, *organism*, *physical force*, *substance*, *specimen*, *social context*, *attribute*, *body structure*, *context-dependent category*, *evens*, *observable entity*, *physical object*, *environment and geographical location*, *qualifier value*, *staging and scales*, *special concepts*, *pharmaceutical-biological product and record artefact*.

SNOMED CT does not exhibit a tree-like taxonomic structure, but neither do the is-a nor the int-sub relations. Every concept of SNOMED CT can be linked hierarchically to exactly one top-level class; this implies that SNOMED CT consists of eighteen independent hierarchies. With respect to the int-sub relation (C is an intensional sub-concept of D) these nodes represent minimal elements, while with respect to the is-a relation they are maximal elements. These hierarchies are associated to different domains.

Bodenreider et al. formulated principles that a good ontology should satisfy and then measured SNOMED against these criteria (Bodenreider et al., 2004b). They first consider classification principles: (a) each hierarchy must have a single root; (b) each class, except for the root, must have at least one parent; (c) non-leaf classes must have at least two children; and (d) each class must have a different definition from all others. In particular, each child must differ from its parent, and siblings must differ from each other. Since a root is a class without a parent by definition and (a) is assumed, the condition (b) follows automatically and need not be stated. Furthermore, condition (d) is automatically satisfied, because concepts (categories) are defined by intentions. Hence, Bodenreider's classification principles can be reduced to: (1) each hierarchy must have single root and (2) non-leaf classes (concepts) must have at least two children.

Bodenreider et al. observed that there are 23,174 single child classes (Bodenreider et al., 2004b). They analyzed the single child concepts and proposed the following classification: (a) the incompleteness of the hierarchy, (b) the presence of a hybrid class, resulting from the intersection of two parent classes as the single child of at least one of the parent classes, and (c) redundant concepts, where a parent and a child concept do not differ. It can easily be seen that case (b) cannot be avoided, and that case (c) is incorrectly analyzed. They claim that the concepts *Closed fracture of skull without intra-cranial injury* and *Closed fracture of skull* represent one biomedical entity. However, the concepts differ, because *Closed fracture of skull without intra-cranial injury* is an aspectual concept derived from the concept *fracture of skull*, which can be understood as an elementary concept.

16.5.3 UMLS

The *Unified Medical Language System* (UMLS) is a project that has been under development at the U.S. National Library of Medicine (NLM) since 1986. The creation of the UMLS was motivated by the increasing number of heterogeneous terminological systems that represent, classify and name the same concepts differently, thus hampering the retrieval and integration of information from different sources. The goal of UMLS is to integrate multiple machine-readable biomedical information sources (e.g. ICD, LOINC, SNOMED, etc.) as transparently as possible.

The UMLS consists of three parts or knowledge sources:

- the Metathesaurus,
- the Semantic Network and
- the SPECIALIST lexicon and other related lexical programs.

The Metathesaurus is the component that ties together the different source vocabularies. A concept in the Metathesaurus links concepts from different source vocabularies that the UMLS editors judge to have the same meaning. An important principle of the Metathesaurus is to preserve the information from the underlying source vocabularies (terms, concepts, codes, attributes, hierarchical and other

relations) in order to achieve transparent integration. Synonym information and other relations between terms and concepts from different vocabularies are added by the UMLS editors in order to achieve this integration. The UMLS combines over 2 million names for more than 900,000 concepts from more than 60 families of biomedical vocabularies, as well as 20 million relational links among these concepts. Vocabularies integrated into the UMLS Metathesaurus include the Gene Ontology, the NCI taxonomy, the Medical Subject Headings (MeSH) and the Digital Anatomist Symbolic Knowledge Base. Some UMLS concepts have also been linked to external resources such as GenBank. The Semantic Network provides a categorization of the concepts contained in the Metathesaurus. The current version of UMLS defines 135 semantic types, which are organized hierarchically (by the is-a relationship) into two distinct hierarchies, rooted by Entity and Event. Each concept is assigned to at least one semantic type. The semantic network is developed independently of the domain specific vocabularies and serves as a basic, high-level ontology for the biomedical domain. It includes 135 semantic types and 54 relationships. Each semantic type in the network has a textual definition and occurs within one of the hierarchies. In addition to the taxonomy, relationships of five subcategories are introduced between semantic types. Since each Metathesaurus concept is associated to at least one semantic type, relationships between semantic types also define the admissible semantics for relationships between concepts.

There are many relational links in the Metathesaurus that originate from the source vocabularies. The UMLS does not directly connect Metathesaurus relational links to Semantic Network relationships. These missing connections demonstrate that UMLS is not semantically founded and coherently organized. This implies, in particular, that the Semantic network relationships cannot be used directly to validate Metathesaurus relationships, as noted in (Vizinor et al., 2006). Furthermore, it is not easy, perhaps even impossible, to transform the UMLS into a semantically founded and coherently organized ontology. But it may be interesting to extract from UMLS semantically founded and coherently organized sub-ontologies, which can then be used for knowledge extensions.

16.5.4 LOINC

LOINC is an acronym for *Logical Observation Identifier Names and Codes*. It has been under development since 1994 in a voluntary effort initiated and coordinated by the Regenstrief Institute for Healthcare in Indianapolis (USA). LOINC's purpose is to facilitate the exchange and reporting of laboratory and other clinical observations, such as blood hemoglobin and serum potassium for clinical care, management and research. Many laboratories and diagnostic services use HL7 to transmit their results electronically from their reporting systems to their receiving systems, which include hospitals and research groups. The Health Level Seven standard (HL7) provides a standardized syntax for specifying messages, but does not employ standardized terms and codes to be used for identifying laboratory tests. LOINC was developed to close this gap: it provides a universal coding system that can be used

in current messaging standards like HL7 for identifying laboratory data and clinical observations, such as blood pressure and temperature.

The *laboratory domain* covered by LOINC not only includes categories of sub-domains such as chemistry, hematology, serology, microbiology and toxicology, but also incorporates categories for drugs and cell counts. LOINC's clinical domain covers categories for vital signs, hemodynamics, EKG, cardiac echo, urologic imaging, pulmonary ventilator management, selected survey instruments and others. LOINC's domain includes as individuals laboratory tests and clinical observations.

LOINC's concepts are based on six axes, aspects or facets. They are *component* (potassium, hemoglobin, hepatitis C antigen), *property measured* (mass concentration, enzyme activity), *timing* (whether the measurement is an observation at a moment of time, or an observation integrated over an extended time), *type of sample* (urine, blood), *type of scale* (whether the measurement is quantitative (true measurement), ordinal (a ranked set of options), nominal (e.g. E. coli) or narrative (e.g. dictation results from x-rays) and *method* (used to produce the result or other observation).

To any of LOINC's 6 axes, denoted by $ax(1), \dots, ax(6)$, domains $D(ax(i))$ can be associated, and for any of these domains $D(ax(i))$ a system of categories/concepts $Cat(ax(i))$ can be introduced. The *domain of scale*, for example, is sub-divided into entities of quantity (numerical values), ordinal quantity (positive, negative, reactive, indeterminate etc.), narrative (texts narratives, such as the description of a microscopic part of a surgical test). Laboratory tests and clinical observations are very complex individuals. But neither all their possible properties are relevant, nor are all their relations useful in the clinical context. Hence, the axes or aspects choose from a possibly infinite number of aspects of tests, only those that are clinically relevant. An individual test is characterized by an aggregate of individual entities that are instances of the aspect categories. This information is collected and represented in a schema with six slots, which differs from other ontologies such as ICD.

16.5.5 GALEN

GALEN is an acronym for *Generalized Architecture for Languages, Encyclopaedias and Nomenclatures in Medicine*. GALEN is a terminological system that provides an entire framework for medical terminology. The GALEN Common Reference Model is geared toward reusable application-independent representation of medical concepts; the basis of GALEN terminology services supports electronic health care records, clinical user faces, classification and coding systems, decision support systems, knowledge management systems and natural language processing. A key feature of GALEN provides a set of building blocks and constraints from which concepts can be composed.

GALEN's development has been influenced by the development of traditional terminologies designed for specific purposes that have an impact on specific organization principles supporting the terminology's intended usage in the chosen axes or basic aspects and the level of granularity in concept representation. GALEN

is designed for developing a framework that is independent in both purpose and application. The Common Reference Model, as an ontology, is formulated in a specialized description logic, called GALEN Representation and Integration Language (GRAIL). This ontology aims to represent all relevant medical concepts, independent of any application (Rector et al., 1997).

A crucial characteristic of GALEN is its construction on the basis of a top level ontology and specific representation formalism. Furthermore, GALEN provides a mechanism for combining simple concepts with more complex ones.⁷ The basic division of the top level categories is between *Phenomenon*, subsuming structures, processes and substances and *Modifier Concept*. The category *Phenomenon* covers entities with independent existences, like physical objects, whereas the category *Modifier* covers dependent entities, including properties, states and roles. Furthermore, GALEN provides a rich system of associative relationships.

16.5.6 MeSH

The MeSH (*Medical Subject Headings*) is a thesaurus published, revised and adapted by the US National Library of Medicine (NML). It is used for cataloguing library holdings and serves as a basis for retrieving bibliographical citations. The basic components of thesauri are descriptors, also called (main) headings, keywords, topics or subjects. According to ISO 2788, they are connected by three relationships “Broader-than” (BT), “Related-to” (RT) and “used-for” (UF). The relationships and other instruments are used to guide and control the choice and combination of descriptors for indexing and searching through information. MeSH, in particular, includes, among others, the following constituents: descriptors, qualifiers, and relations. Ingenerf et al. claim that the main purpose of MeSH is epistemology, rather than ontology, and that ontological principles should be applied to thesauri with great caution (Ingenerf and Lindner, 2006). They analyzed MeSH with respect to certain ontological principles related to the is-a relation and to epistemology. These principles state that overloading the is-a relation and allowing epistemology to intrude must be avoided. They then claim that a rigorous application of these principles to terminological systems, including the elimination of epistemology-loaded terms, will increase the mismatch between purified ontologies and epistemologically “infected” terminology systems.⁸

⁷It seems that these principles cover only a tiny fragment of rules for concept formation, compared with the vast body of knowledge about concept formation available in cognitive linguistics and cognitive psychology. In particular, the description logic formalism includes only a small fragment of predicate logic, and predicate logic provides only a small fragment of concept formation principles expressible in other formal languages.

⁸One can imagine a situation in which only purified ontologies survive, and the epistemologically spoiled terminological systems, together with the mismatch problem, disappear from the scene. There are proposals whose realization would lead to such a consequence. In Smith et al. (2005), for example, the authors criticize the work of the ISO Technical committee and claim that it produces

We show that the mismatch problem is not caused by ontology in general, but is a consequence of an overly narrow view of a particular kind of philosophical realism and a restricted understanding of a top level ontology. The GFO approach, which provides several kinds of categories, higher order categories and a theory of levels, may cope with these problems. Outlined below are some ideas regarding how the MeSH constituents can be ontologically reconstructed in the GFO-framework using the theory set forth in this paper. The analyses are restricted to the new notion of *descriptor* and to the *broader-than* relation. A complete ontological reconstruction will be published elsewhere.

Nelson et al. propose a new structure for MeSH that centers on descriptors, concepts, and terms (Nelson et al., 2000). A descriptor is defined as a class of concepts, and a concept as a class of synonymous terms within a descriptor class. Such a descriptor (Descr) can be considered as a higher order concept, whose instances are the concepts within a class. There are preferred concepts with a selected term that serves to denote the (higher order) concept Descr. The concept Descr itself is specified by information related to instances and their inter-relations. A concept in Descr is denoted by a set of synonymous terms. As instances of Descr, the concepts stand in relation to the preferred concept (narrower, broader, related). The descriptor's name is used for any of its instances.

The hierarchical relationships in MeSH are implicitly included in the "broader than" relationship BT. The BT relation has many meanings, among them "is-a" (extensional subsumption), "part-of" (mereology), "conceptual-part-of" (resembling the relation categorial-part-of) and "process-of". If "is-a" is the primary meaning, then the phenomenon of is-a-overloading occurs, which must be, according to the principles of ontological purity, excluded (Guarino and Welty, 2002). But the BT relation cannot be reduced to the before-mentioned relations, because another important facet must be considered, namely the search for documents. Soergel defines the BT relation in a manner considered to capture this relation's nature (Soergel, 1985). *Should a search for documents be dealing with (the term/concept/descriptor) A find all (or most) documents dealing with (the term/concept/ descriptor) B? If yes, then A is broader than B (and conversely, B is narrower than A).*

This definition is, of course, not an explicit mathematical definition; nonetheless, it can be described in terms of higher-order concepts that are included in GFO. Let D be a document and $C(D)$ a higher-order concept having all concepts occurring in D as instances. Furthermore, let SetDok be a fixed set of documents. Then the above relation may approximately be described as follows: a descriptor c is broader than the descriptor d with respect to the set SetDok , if for all D in SetDok the

weak results inherited from the earlier work of the ISO TC 37. The ISO TC 37 was influenced by a "certain Eugen Wüster (1898–1977), an Austrian businessman, saw-manufacturer,...., and devotee of Esperanto" (Smith et al., 2005). This "saw-manufacturer" and "devotee of Esperanto" is, in the opinion of Smith et al., responsible for an aberration of terminology research that hampers the development of purified ontologies.

condition $d::D$ implies $e::D$. The relation BT depends, obviously, on the reference set of documents $\text{SetDo } k$.⁹

In many cases the relation BT represents the taxonomic is-a relation, for example “Pain BT Abdominal Pain”, in others not, for example “Abdomen BT Stomach”. MeSH’s ontological foundation includes an analysis and explication of those relations implicitly contained in BT, which present relations between primitive concepts.¹⁰ Such an analysis is a necessary presupposition for solving the matching problem between MeSH and other medical terminological systems. Furthermore, it seems to be an interesting task, as a part of the ontological analysis, to extract a maximal taxonomic tree from the MeSH thesaurus. The 16 broadest headings of MeSH include, among others, *anatomy, organisms, diseases, chemicals and drugs, psychiatry and psychology, biological sciences, humanities, information sciences, named group and, geography.*

16.6 Conclusions and Future Research

The field of terminologies is complex and needs to include elements from several sources, including philosophical ontology, linguistics and logic, artificial intelligence. Ontologies may help to clarify the most general principles for building terminologies. The approach presented here admits several ontological types of categories. Furthermore, it also introduces, according to a domain, several kinds of categories, called principal categories, elementary categories and aspectual categories; these exhibit a system which we introduce as a graduated conceptualization. The principal and elementary categories establish the backbone of the ontology of a domain. An open-ended number of aspectual categories may then be added. Between the categories certain basic relations hold; the most basic are the is-a relation and the categorial-part relation. The categorial-part relation addresses the intention of a category; the is-a-relation is associated with the categories’ extension. Furthermore, a number of domain specific relations between categories must be added to the ontology. To develop the next generation of medical ontologies, additional topics, we believe, must be investigated, which are listed below.

- (1) *Integration Schemata and Coherent Organization of Ontologies.* It seems to be useful and efficient to introduce an integrated view on a domain by taking into account the sub-domains of the (main) domain and their representation by using higher-order categories. For this purpose a new type of integration schema must be introduced that allows for a systematic unification of distributed, scattered information and knowledge about a domain. This needs a deeper understanding of the architecture of concepts.

⁹A more accurate definition must consider the distinction between terms and concepts. Furthermore, this definition has a relative nature, because it depends on a reference set of documents. Also, one must clarify whether the BT relation behaves monotonically with respect to the reference set SetDok . However, this does not appear to be true.

¹⁰A concept is said to be primitive if its instances are individuals.

- (2) *Knowledge Extensions of Graph-based Ontologies.* Most ontologies hitherto are presented in a graph-based form; in these ontologies the only axioms are presented as relational links between concepts. A knowledge extension of such an ontology adds more expressive axioms to it. Such knowledge extensions can be used to solve problems and create hypotheses in biology, biomedicine, and medicine. We hold that a sound knowledge extension must be based on coherently organized and semantically founded ontology.
- (3) *Default Knowledge and Non-Monotonic Reasoning.* The integration of the diverse bio-medical ontologies needs a broader framework that allows consistent inferences. One of the severe unresolved problems is concerned with integrating ontologies that present default knowledge with ontologies containing exceptions and deviations from idealized cases. To overcome these problems, principles of non-monotonic reasoning must be introduced. Existing non-monotonic systems, studied in artificial intelligence (McCarthy, 1980, 1986; Reiter, 1980), do not suffice to capture the diversity of situations that occur in life sciences. A first contribution for establishing the basis for such framework is expounded in (Hoehndorf et al., 2007).
- (4) *Ontology of higher order concepts and their architecture.* The structure and formal representation of concepts, notably of higher order concepts, are not yet sufficiently understood. An integration schema of natural concepts, for example, can be understood as concept representation. A concept can be unfolded by adding aspectual categories, derived from it. Aspectual categories of a category add further information and show how the instances of concepts can be further structured.
- (5) *Ontology of Levels of Reality.* The theory of levels is one of the most difficult unresolved problems in ontology. A number of open problems pertains to levels, among them the ontology of multiple inheritance taxonomies, the clarification of views and the ontology of multiple view domains, as well as a deeper understanding of classification principles.
- (6) *Computer-based Ontology-Tools.* An important idea is collaborative ontology development. This is one of the most active areas of research today.

Acknowledgments Many thanks to Frank Loebe, Robert Hoehndorf, Roberto Poli, Josef Ingenerf, Janet Kelso, Jörg Niggemann, Matthew West, and anonymous reviewers for their critical remarks that contribute to the quality of the paper. I am grateful to Dayana Goldstein for her attentive reading which led to an improvement of the text. Many thanks to Christine Green for her help in preparing the English manuscript.

References

- Albertazzi, L. 2001. Presentational primitives. Parts, wholes and psychophysics. In *Early European Contributors to Cognitive Science*, ed. L. Albertazzi, 29–60. Netherlands: Kluwer.
- Albertazzi, L. 2003. From kanizsa back to benussi: Varieties of intentional, reference, *Axiomathes*, 13:3–4, 239–259.
- Bodenreider, F., and R. Stevens. 2006 Bio-ontologies: Current trends and future directions. *Briefing in Bioinformatics* 7(3):256–274.

- Bodenreider, O., B. Smith, and A. Burgun. 2004a. The ontology-epistemology divide: A case study in medical terminology. *Proceedings of FOIS 2004*.
- Bodenreider, O., B. Smith, A. Kumar, and A. Burgun. 2004b. Investigating subsumption in DL-based terminologies. A case study in SNOMED CT. In *Proceedings of the First International Workshop on Formal Biomedical Knowledge Representation*, eds. U. Hahn, S. Schulz, and R. Cornet, 12–20.
- Campbell, K.E., D.E. Oliver, K.A. Spackman, E.H. Shortliffe. 1998. Representing words, and things in the UMLS. *Journal of the American Medical Information Association* 5:421–431.
- Cohen, H., and C. Lefebvre. 2005. *Handbook of categorization in cognitive science*, eds. H. Cohen, and C. Lefebvre. Oxford: Elsevier.
- de Keizer, N.F., and A. Abu-Hanna. 2000. Understanding terminological systems. II: Experience with conceptual and formal representation of structure. *Methods of Information in Medicine* 39:22–29.
- de Keizer, N.F., A. Abu-Hanna, and J.H.M. Zwetsloot-Schonk. 2000. Understanding terminological systems. I: terminology and typology. *Methods of Information in Medicine* 39:16–21
- Gruber, T.R. 1993. A translation approach to portable ontologies. *Knowledge Acquisition* 5:99–220.
- Guarino, N., and C.A. Welty. 2002. Evaluating ontological decisions with ontoclean. *Communications of the ACM* 45(2),61–65.
- Hennig, W. 1950. *Grundzüge einer theorie der phylogenetischen systematik*. Berlin: Deutscher Zentralverlag.
- Hennig, W. 1975. Cladistic analysis or cladistic classification. *Systematic Zoology* 24:244–256.
- Herre, H. et al., 2006. General formal ontology (GFO) – A foundational ontology integrating objects and processes, Part I: Basic principles, *Onto-Med Report* 8 ISSN
- Herre, H. 2010. General formal ontology – A foundational ontology for conceptual modeling, this volume.
- Hoehndorf, F. J. Loebe H. Kelso. 2007. Herre representing default knowledge in biomedical ontologies: Application to the integration of anatomy and phenotype ontologies, *BMC Bioinformatics* 8:377. doi:10.1186/1471-2105-8-377.
- Ingenierf, J., and R. Lindner. 2006. Ontological principles applied to biomedical vocabularies. In *Proceedings of the EFMI Special Topic Conference Integrating Biomedical Information: From Cell to Patient*, eds. Reichert, A. et al., 319–334. Berlin: AKA-Verlag.
- McCarthy, J. 1980. Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence* 13:27–39.
- McCarthy, J. 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 28:89–116.
- McCray, A.T. 2006. Conceptualizing the world: Lessons from history. *Journal of Biomedical Informatics* 39:267–273.
- Nelson, S.J., M. Schopen, J.-L. Schulman, and N. Arluk. 2000. An interlingual database of MeSH translations. In *Proceedings of the 8th International Congress of Medical Librarianship*, London.
- Poli, R. 2006. Levels of reality and the psychological stratum. *Revue Internationale de Philosophie* 61(2):163–180.
- Ranganathan, S.R. 1962. *Prolegomena of library classification*. Bombay: Asia Publishing House.
- Rector, A.L., S. Bechhofer, C.A. Goble, I. Horrocks, W.A. Nowlan, and W. D. Solomon. 1997. The GRAIL concept modeling language for medical terminology. *Artificial Intelligence in Medicine* 9(2):139–171.
- Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.
- Rosch, E. 1975. Cognitive reference points. *Cognitive Psychology* 7:532–547.
- Smith, B. 2004. Beyond concepts: Ontology as reality representation. In: *FOIS, International Conference on Formal Ontology and Information Systems*. Turin.
- Smith, B., W. Ceusters, and R. Temmermann. 2005. Wüsteria. In: *Proceedings Medical Informatics Europe 2005, Geneva; Stud Health Technol Inform* 116:647–652.

- Smith, B. et al., 2007. The OBO foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol* 25(11):1251–1255.
- Soergel, D. 1985. Organizing information. FL: Academic Press.
- Vizitor, L., O. Bodenreider, L. Peters, and A.T. McCray. 2006. Enhancing biomedical ontologies through alignment of semantic relationships: Explortory approaches. *Proceedings of AMIA Annual Symposium* 2006:804–808.
- Wertheimer, M. 1912. Experimentelle Studien über das Sehen von Bewegung. *Zietschrift für psychologie* 1912:161–265.
- Wiley, E.O. 1981. Phylogenetics: The theory and practice of phylogenetic systems. Newyork, NY: Wiley.

Chapter 17

Ontologies of Language and Language Processing

John A. Bateman

17.1 Introduction

Automatic language processing is a particularly knowledge-intensive enterprise. Understanding natural language texts, producing such texts, finding the most important information in a collection of texts for the purposes of summarization or information extraction, and translating texts from one language to another are all examples of natural language processing applications that benefit from a detailed representation of what is being communicated (cf. Dale et al., 2000). For this reason, language processing and knowledge representation are regularly found in combination and there is a long history of interaction. The need for explicit domain representation also overlaps with the task of constructing domain ontologies: clearly, one way of making available information about the content of some collection of potential texts is to draw on ontologies of the domains treated in those texts.

This leads to two inter-related problems. First, in the vast majority of areas where natural language processing is attempted, there are no ready made domain ontologies available for use. This is the problem of *breadth* – natural language processing typically calls for a very broad coverage of commonsense and specialised domains of application. Second, even when there are domain ontologies available, it is by no means straightforward to relate those ontologies to the kinds of knowledge organisation supportive of natural language processing. This is the problem of *mediation* between ontological organisations motivated by domain and application and organisations motivated by the needs of expressing that information in natural language. These distinct purposes do not necessarily lead to organizational structures that align.

The relationship of natural language processing to ontological engineering is therefore complex and makes contact with several central theoretical issues of current concern. In this chapter, we set out those approaches where the two areas of research – linguistic processing and ontology – come the closest and where

J.A. Bateman (✉)
University of Bremen, Bremen, Germany
e-mail: bateman@uni-bremen.de

current research involves both areas in combination. We will see that some quite diverse organizational assumptions are made by the accounts that we consider. This is because they represent different starting points, different methodologies, different purposes and different criteria for success. Comparing such diverse approaches requires us to be aware of these differences; the distinct kinds of information contributed may all be useful for the overall task but bringing them together is not straightforward. Approaches to achieving such combinations vary in sophistication and there are many open questions. In many respects even the use of the term “ontology” in this domain is not uncontroversial. There is considerable overlap between the approaches we discuss and traditional work on lexical databases and machine-readable dictionaries, where the ontological import of what is being described is at best debatable. The precise nature of the information contributed from linguistically oriented ontological development therefore needs to be carefully considered.

The distinctive feature shared by all the frameworks that we discuss is that they are intended to capture significant aspects of the organisation of natural language. Just how they set about this depends on both philosophical orientation and practical demands. We find influences from traditional linguistic positions concerning the relation between language and world, from psychological considerations of the relation between language and its processing by the human brain, from semiotic considerations of the nature of signs, from computational considerations of how best to construct broad-scale lexicons, and many others. However, since the purpose of this chapter is primarily to show linguistic processing as an “application domain” for ontology, we will not delve more deeply into philosophical positions taken on the ontology of language-as-such (cf. Pease and Niles, 2002; Bateman, 2004). Moreover, some of the approaches we deal with have also been introduced in other chapters of these volumes: for these, therefore, our focus will be on bringing out more clearly their relationships with the other accounts in the area that we discuss.

We will characterize the different approaches to ontology and language in terms of the status they assign to *linguistic semantics*. This is also the clearest point of contact with the variety of philosophical positions taken on language – particularly in regard to the relationship between language and the world. If we consider any example sentence, this can generally be seen as picking out some state of affairs in the world. But how this relationship is theorised more precisely opens up space for several alternative treatments.

If, for example, the sentence is taken to refer “directly” to its corresponding state of affairs, then we have a view where the meaning of the sentence is best articulated in the same terms that are used to describe such states of affairs in general. Formulating linguistic semantics then overlaps to a considerable extent with the tasks of constructing individual *domain ontologies*. This perspective sees little difference between semantics and knowledge engineering for a domain and is, accordingly, most commonly found in approaches driven by Artificial Intelligence or Computer Science. It also occurs in some approaches to language that start from ontology. However, there is strong linguistic evidence that these kinds of

information should be held apart: conflating them is superficially plausible but only for relatively unsophisticated uses of language and language processing.

If the sentence is then taken to create some additional level of semantic representation that must first be *related* to its corresponding states of affairs for the meaning to be found, then we begin to see a distinct contribution from linguistic semantics over and above that provided by the modelling of some domain. Most sophisticated views of language adopt some variant of this view. In general, a context-free semantic representation corresponding to the linguistic expression at issue needs to be *contextualized* to provide a context-dependent interpretation: i.e. one in which, for example, the concrete referents corresponding to linguistically open expressions such as “I”, “you”, “this box” and so on have been identified. This simplifies the specification of a compositional semantics for linguistic expressions and gives a firm place for contextualisation procedures to start. In more recent approaches, the range of linguistic expressions for which this kind of two-stage contextualisation is found useful has grown considerably.

Several distinct kinds of approach can be recognised according to how this relation between linguistic semantics and domain is conceived. At one extreme, little difference in kind is seen between the representation of linguistic semantics and representations of domain knowledge apart from the degree of specificity: linguistic semantic representations are then “under-specified” versions of their contextualized interpretations (cf. van Deemter and Peter, 1996). At the other extreme, there need be no particular relationship between the two levels of representation at all: they are just placed in “correspondence”. Stone (2003a), for example, proposes that linguistic semantics provide *descriptions* of domain states of affairs but do not correspond to them more directly.

This latter position leads to the final bifurcation that we will build on. If we have two relevant levels of representation, the linguistic semantics and the domain model, then we can ask about the connection between ontology and *each* of these levels separately. This gives rise to a further range of approaches: at one extreme, the linguistic semantics is not given anything resembling an ontological treatment while, at the other, ontological methods are applied for linguistic semantics also (as done, for example, each from a rather different perspective, by Bateman et al., 1995a; Nirenburg and Raskin, 2001; Cimiano and Reyle, 2006). Debate continues both about which of these possible approaches to the relationship between language and world is the most appropriate and about whether this has anything to do with ontology.

The three positions that we have set out are summarised graphically in Fig. 17.1. Model (I) is the situation in which the meaning of linguistic expressions is nothing other than the representation of some domain state of affairs; model (II) interposes some semantic organisation that is distinct from the domain state of affairs but which is not strongly structured in its own right; and model (III) finally assumes that both linguistic semantics and domain representations may be approached ontologically and that their contributions are distinct from one another. With these three positions established, it is rather more straightforward to characterize the diverse positions on language, language processing and ontology currently being pursued.

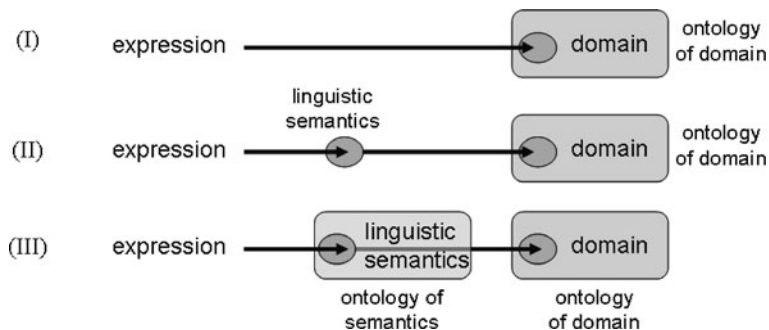


Fig. 17.1 Three different positions taken with respect to language, the world, and the role of linguistic semantics in mediating between them

17.2 Lexical Databases and Ontology

Much of linguistic thinking at present is dominated by the lexicon. Essentially the lexicon is the repository of meaningful units that may be employed in a language and which need to be learnt rather than derived by more general principles, such as the combinatorial rules of syntax. It is evident, however, that the lexicon of any language is far more than simply a list of form-meaning pairs. There is considerable organisation within the lexicon and so a major goal remains to capture sources of generalisation and productivity so as both to make more accurate statements and predictions concerning how people use their lexical stock and to achieve an appropriately strong theoretical position on just what the organisation of a lexicon can be. There are accordingly many large-scale “lexical database” projects that are attempting to document the range of lexical material available in various languages. In general, the relationship of this work to questions of ontology is at best indirect, but there are also approaches, particularly those conforming to model (I) in Fig. 17.1, that invite a conflation of concerns in which the line between work on lexical organisation and work on domain ontologies becomes quite blurred.

Early psychological work on how people use words provided evidence that there is a strong *associative* aspect to lexical organisation. Given a particular word, an experimental participant is far more likely to name certain associated words rather than others. Moreover, within traditional semantics there had also been characterizations of just what kinds of associations there might be. These are usually termed *lexical semantic relations* (Cruse, 1986) and encompass such relationships as *hyponymy* (an “is-a” relationship), *meronymy* (a “part-of” relation), *synonymy*, *antonymy* and so on. This kind of organisation forms the basis for the development of the original Princeton WordNet for English (cf. Chapter 10). Words (or rather, word senses) that were considered synonyms were grouped into collections called *synsets* so that the members of a synset mutually disambiguate one another. Lexical semantic relations are then defined between synsets and this gives rise to a net-like lexical organisation resembling a multidimensional thesaurus. The main descriptive work covered by WordNet was initially its treatment of nouns and verbs, although now the coverage

of adjectives is also quite broad; the current WordNet version (3.0 at time of writing) contains 81,426 noun synsets, 13,650 verb synsets and 18,877 adjective synsets.

Although the developers of WordNet generally refer to it as a lexical-semantic net or lexical database, others commonly use the term *lexical ontology* or *lexical-semantic ontology*. This usage rests on the similarity between aspects of WordNet's organisation and that typically found in lightweight domain ontologies: i.e. a class-subclass inheritance backbone. Then, since the WordNet organisation often provides information in areas for which there is no domain ontology, it appears a natural candidate to take on this role. This has been most useful in applications that need a notion of "related" words without a deep axiomatisation of the meanings involved – for example, in information retrieval, where simply expanding a query to include a disjunction of the words in a synset (and its superclasses) can increase the range of documents retrieved substantially. It has also been suggested for use in enriching Semantic Web ontologies (cf. [Chapter 9](#)), which again are generally constrained to be lightweight with respect to their axiomatisation.

The loose axiomatisation results, however, in a structure for which it is difficult to gauge the accuracy of the statements that it entails concerning the elements it relates. The "upper" part of its class-subclass hierarchies fails to capture a number of generalisations and relationships that would be expected and so reduces the value of the organisation as a whole when inferences are to be performed. To solve this problem there have been several attempts to augment the WordNet lexical information with a stronger semantics. One of these is **OntoWordNet** (Gangemi et al., 2002), which replaces the upper parts of the noun taxonomy with categories drawn from the DOLCE (cf. [Chapter 13](#)) ontology, and another is the combination of SUMO (cf. [Chapter 11](#)) with WordNet (Niles and Pease, 2003). This has also been attempted in some of the multilingual versions of WordNet, particularly EuroWordNet, which has its own "upper" organisation (Vossen et al., 1997). There are some indications that the problems go deeper than simply adjusting the inter-relationships among the top level classes, however. Trautwein and Grenon (2004), for example, argue that dealing with *role* information adequately from an ontological perspective will require more substantial changes. Again we need here to consider just why a lexically motivated organisation should benefit from the foundational ontologies proposed: this only follows as a consequence of model (I) above where the boundaries between lexical and ontological information is unclear. Nevertheless, there are also strong arguments that an explicit orientation to ontological principles is beneficial when designing lexical resources, for example to avoid the well-known knowledge representation problem of "*is-a*"-overloading (cf. Guarino, 1998).

A further framework that characterizes itself as describing lexical semantics but which also appears as a candidate for domain ontologies of various kinds is that of **FrameNet** (Baker et al., 1998; Fillmore and Atkins, 1998). FrameNet is a development of Fillmore's (1976) notion of *frame semantics*, in which the semantics of lexical units is captured by associating within single bundles, termed *frames*, some identifiable action or situation and those roles, termed *frame elements*, that regularly participate in that action or situation. Lexical units are said to "evoke" frames. FrameNet is also being used to annotate corpora of naturally occurring texts: this

proceeds by finding occurrences of head lexical items and then annotating accompanying linguistic elements with their corresponding frame element roles. An example of such an annotated expression is the following:

[Cook Matilde]**fried**[Food the catfish][Heating_instrument in a heavy iron skillet]

Here the frame elements (roles) are given as subscripts to the phrases expressing them; the entire example is an illustration of the COOKING CREATION frame.

Frames are organised into an inheritance hierarchy where child frames inherit the frame elements of their parents. The COOKING CREATION frame, for example, inherits from the INTENTIONALLY CREATE frame (with frame elements “creator” and “created entity”), which in turn inherits from CREATING and INTENTIONALLY ACT (which brings in an “agent” frame element). However, since frame elements range across roles that are specific to particular frames (e.g. the “defendant” frame element in a VERDICT frame) and quite general roles (e.g. agent), the relation between frame elements of frames standing in inheritance relationships is not straightforward. Probably partly as a consequence of this, the FrameNet hierarchy is relatively shallow.

A hierarchy of additional *semantic types* defines categories that appear to run orthogonally to the frame hierarchy. Two examples of such types are “sentient” and “container”. The FrameNet developers suggest that these can be related straightforwardly both to WordNet synsets and to “ontological” categories, although they also note that there is no guarantee that the precise class-subclass relations will be found in any other resource in quite the same way. As usual, there are many fine-grained differences between the particular reading of these categories given in FrameNet and any particular definition in other ontologies, which suggests that the categories involved have not yet been sufficiently understood.

There are also a variety of further relationships defined between frames. For example, a TRANSFER SCENARIO frame defines a situation in which an entity playing the role of the frame element “donor” transfers an entity playing the role of a “theme” to an entity playing the role of a “recipient”. This frame is related by *perspectivalisation* to the frames of GIVING SCENARIO and RECEIVING SCENARIO. The former then combines with INTENTIONALLY ACT to yield the GIVING frame that most directly corresponds (or is evoked by) sentences with the verb “give” in them. There is little explicit representation of the precise semantic consequences involved in these inter-frame relationships however; the task remains lexicographic rather than that of providing a formal semantics for linguistic expressions that might support reasoning. It is up to the human interpreter to understand the distinct annotations and glosses of particular frames that are defined.

Nevertheless, it was in early knowledge representation systems (Minsky, 1975) that the notion of frames was first employed and, in certain respects, FrameNet still resembles what one would expect in a generic domain model or ontology. In particular, certain relations defined between frames allow complex situations to be described with internal structure made up of contributing subframes. An example of this is the CRIMINAL PROCESS frame, which has subframes of ARREST,

ARRAIGNMENT, TRIAL and SENTENCING. This “script-like” use of frames clearly goes well beyond what would be expected of a straightforward lexical database and also contributes to readings of FrameNet as another kind of rather general domain ontology.

But, as with WordNet, the accuracy both of the hierarchy of frames that is proposed and their interrelationships requires substantially more theoretical and empirical verification. The motivations for distinguishing frame elements are essentially lexicographical with corpus support. This means that frames are proposed on the basis of intuition and then actual occurrences of lexical items are checked in corpora to see whether the frame elements proposed for the class are sufficient. Appeal is made to grammatical distinctions, i.e. whether classes have similar selection restrictions, but the developers appear willing to deviate from any particular grammatical evidence if their semantic intuition suggests otherwise. In order to improve on this situation and to provide inferential capabilities, Burchardt and Frank (2005) report on an experiment in text interpretation where FrameNet frames are linked with SUMO (cf. Chapter 11) to provide some of the missing semantic axiomatisation. Far more work of this kind will be necessary to bring out to what extent the hierarchical relationships proposed in FrameNet do in fact correspond to formalizations useful for domain modelling.

The current version of FrameNet (May 2010) covers over 10,000 lexical units involving more than 1014 semantic frames, exemplified in more than 135,000 annotated sentences. The explicit link that FrameNet provides between natural examples of linguistic expressions and frames and frame elements provided by FrameNet therefore provides an excellent position from which to start validation.

17.3 Grammatical Motivation and Linguistic Ontology

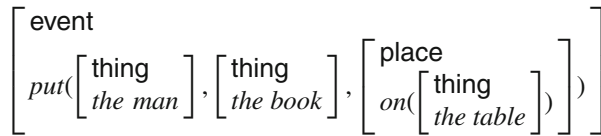
Whereas the original WordNet contained very little information concerning the *syntactic* properties of its elements, such information is usually an important part of lexical organisation. That is, it is important not only to know how words are related lexically-semanticly, but also to know in which kinds of grammatical contexts they can be employed. Information of this kind is included within FrameNet, particularly in the form of the annotated corpus examples, but does not play a strong role in the motivation of categories and relationships between frames.

A further range of approaches to capturing semantic organisations entailed by language take a rather different position and include grammatical evidence as one strong source of motivation for the organisations constructed. This builds on a basic premise of many modern schools of linguistics: that is, that regularities in syntactic distribution are indicative, or symptomatic, of underlying commonalities in semantic classes. Here accounts draw on a variety of syntactic evidence and use this to motivate semantic categories of various kinds.

One early example of this is the lexico-conceptual structure proposed by Jackendoff (1983). Jackendoff states that:

... every major phrasal constituent in the syntax of a sentence corresponds to a conceptual constituent that belongs to one of the major ontological categories. (Jackendoff, 1983, p. 67)

The following is then an approximation to the conceptual structure for the sentence *The man put the book on the table* (Jackendoff, 1983, p. 68):



The structure may be glossed as stating that a predicate *put* of type *event* holds over three arguments: the first two are of type *thing*, the latter is an *on*-relation of type *place*. Each of the predicates are taken to be defined as semantico-conceptual categories motivated primarily by linguistic patterning. This corresponds most closely to model (II) in Fig. 17.1 since the semantics and the domain representation clearly overlap but begin to show distinct organisations.

Further examples of the motivation of semantico-conceptual categories from linguistic evidence can be seen in the following list of categories offered by Jackendoff:

	<i>Interrogative probe</i>	<i>supports category:</i>
a.	What did you buy?	[thing]
b.	Where is my coat?	[place]
c.	Where did they go?	[direction]
d.	What did you do?	[action]
e.	What happened next?	[event]
f.	How did you cook the eggs?	[event]
g.	How long was the fish?	[amount]

Subsequently, further categories of differentiations are made working from intuitions concerning the meanings of sentences and their constituents supported by example sentences. Particular co-occurrence regularities over these categories are then defined by well-formedness rules and this goes some way towards establishing a hierarchy of interrelated categories analogous to the standard hierarchical organisations we see in ontology construction. A similar but different set of categories is motivated in an identical manner by Talmy (1987), particularly for spatial meanings. By considering how a range of languages construct their views of space, Talmy is led to the following characterisation of the distinct types of entities that can be located in space:

dimension	continuous	discrete
space :	matter	objects
time :	action	events

Further distinctions Talmy motivates are *plexity*, a generalisation of singular and plural to include events, and *boundedness*. The approach as a whole is set out in considerable detail in Talmy (2000, 2006).

Although in these classifications we can observe similarities to categories found in other ontological approaches, it is important to note that these organisations are driven by linguistic facts rather than by a consideration of how the world must be – i.e., the evidence and way of reasoning is explicitly linguistic rather than world-oriented and ontological. Several researchers have produced organisations, or ontologies, based on linguistic evidence in this way (cf. Langacker, 1987; Wierzbicka, 1988; Frawley, 1992; Dahlgren, 1995). As Talmy, Halliday and Matthiessen, and others note (see below), open-class lexical distinctions are not so revealing here and so the organisations that result are largely dependent on the breadth of *grammatical* information considered. Whereas some approaches aim at broad descriptions of the semantics of language over all, others focus on quite specific areas in detail: e.g. Asher's (1993) and others' investigation of "abstract objects" such as *facts* and *propositions*, Bierwisch and Lang's (1989) consideration of dimensional information, Vendler's (1957) analysis of the temporal profiles of events and states, Davidson's (1967) and Parsons's (1990) analysis of the internal structure of "events", and many more. One common aspect for many of the "ontological" characterisations motivated from linguistic sources, however, is the primacy given to events and objects. Information is generally centered around an event frame or configuration, and objects are related to this single entity by binary role relations. Although this in fact directly follows from natural language semantics, it is also regularly, and often independently in separate disciplines, proposed as an effective organisation for knowledge representation as such (e.g., Bateman, 1990; Park, 1995; Kuhn, 2001; Stone, 2003a; Zarri, 2005); debate on the issue continues.

The use of linguistic evidence rests on the basic principle that if a distinction is drawn in language, then it may be beneficial to consider this for its ontological import also. This then goes considerably beyond populating domain ontologies with the entities that are demarcated by lexical items. For example, a description of the lexical field of furniture – tables, chairs and so on – will probably differ little from that which a commonsense domain ontology of furniture would deliver. Since the domain of furniture is a socially constructed set of artefacts maintained largely by the explicit naming conventions established by its society of users, there are few places to look for evidence of its structure other than the associated lexical items and their usage in language (although there are also difficult questions here to be raised about precisely *whose* language usage is relevant). Distinctions of this kind can be invented more or less at will and it is unlikely that they raise significant issues for ontology. They will be described by some bundle or configuration of properties that are ontologically grounded without themselves effecting that ground.

The import of *grammatical* organisation, rather than lexical distinctions of the kind seen in WordNet and FrameNet, is very different. In contrast to the more or less arbitrary distinctions accumulated in the lexicon, the grammar of a language is a complex system that has evolved over time to meet *all* representational requirements of its users. As such it can be considered as a culture's "theory" of its world and

applies to all meaning-making that that culture undertakes (with language). This is necessarily far less susceptible to arbitrariness (cf. Halliday and Matthiessen, 1999; Talmy, 2000) and may well have a sufficiently strong influence on our construction of reality as to warrant ontological consideration.

This sets up a useful contrast between lexically-based developments, such as that of WordNet and FrameNet, and a further range of ‘linguistically-motivated’ ontologies where it is grammatical evidence that takes on the role of the prime source of evidence for the categories proposed. Of these, closest to FrameNet is probably **VerbNet** (Kipper-Schuler 2005), an extension of the original work by Levin (1993) on verb *alternations*. Alternation is a linguistic relationship defined over contrasting but similar grammatical constructions. The most well known example is the dative alternation illustrated in the following pair:

- (a) She gave a book to the man
- (b) She gave the man a book

There are a substantial number of such alternations but they do not apply uniformly to all clauses. Levin’s premise was that the particular set of alternations that any particular *class of verbs* undergoes is an indication of an underlying semantic similarity that all those verbs share. That is: verbs that undergo dative shift may be distinguished semantically from those that do not. Mapping the alternations of a language is then one way of discovering the implicit semantic categories involved: these categories are not subject to definition and discussion by language users in the same way that, for example, the distinction between “chair” and “sofa” might be, but instead form part of the implicit (for its users) meaning-making potential of the language itself.

On the basis of the classes proposed by Levin and several further alternations that have been investigated since, VerbNet distinguishes 237 distinct verb classes for English. Each class includes information concerning the grammatical constructions possible for its members, the semantic restrictions holding for grammatical constituents (e.g. whether a constituent must express an “animate” entity or not), and a skeleton semantics in terms of configurations drawn from around 90 semantic predicates. As with FrameNet, the constituents associated within any particular class are defined in terms of so-called *thematic roles* such as “agent”, “recipient”, etc. In contrast to FrameNet, however, the roles are taken from a fixed set of generic modes of participation and are not specific to particular classes. The result is again something like a lexicon of verbs, with a range of semantic categories associated with those lexical items, but the organisation of this lexicon is assumed to be semantically motivated. The semantic predicates invoked here are not themselves well organised, however, and so the approach as such conforms to model (II) in Fig. 17.1, i.e. a semantics lacking strong organisation.

VerbNet’s developers believe that VerbNet will prove more reliable and helpful in automatic processing than efforts such as FrameNet precisely because of the explicit link that is maintained with grammatical motivation. In addition, VerbNet classes include specifications of the grammatical contexts in which they may appear, expressed in terms of the extremely detailed XTAG grammar of English (Doran et al., 1994): this is considerably more detailed than the looser lexicographically

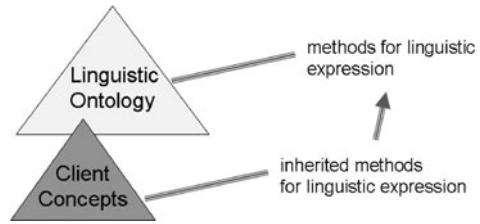
inspired syntactic characterizations found in FrameNet. In opposition to this, however, Baker and Ruppenhofer (2002) argue that FrameNet will produce better results because it does not force apart lexical items simply because they do not share alternation patterns: FrameNet allows forms with quite varied syntactic properties to be grouped together as members of single frames whenever its developers believe that the semantic relation is strong enough. But this may have as a consequence that its categories are not sufficiently tied to linguistic realisation, which may render it *more* difficult to employ for automatic processing, not less.

Another approach where grammatical motivation is considered fundamental is the **Ideation Base** of Halliday and Matthiessen (1999). This is the broadest “linguistically motivated” ontology in terms of the phenomena addressed, although it is not equally developed in all areas. Whereas VerbNet extends the original approach of Levin by considering more of a verb’s syntactic “context” than just noun phrase and prepositional phrase complements, the Ideation Base goes further and accepts grammatical evidence of any kind, regardless of where in the grammar it is to be found. The resulting organisation is not then a lexicon of any particular grammatical classes – such as verbs in VerbNet’s case – but is itself a linguistically motivated organisation of the underlying semantic predicates required. This organisation is then itself considered to be subject to ontological principles and so moves the entire account into our model (III) in Fig. 17.1.

The Ideation Base is being developed further as a computational ontology called the **Generalized Upper Model** (GUM: Bateman et al., 1995a, 1995b, 2008) which is itself descended from the PENMAN Upper Model (Mann et al., 1985, Bateman, 1990), a domain- and task-independent knowledge organisation originally used to mediate between domain knowledge and the Penman natural language text generation system. For this system it was necessary to support a form of modularity by which the linguistic components of the generation system could be “insulated” from any specific knowledge modelling commitments made in individual domains – otherwise the system would not itself have been domain-independent. This was achieved by placing the Upper Model “between” domain model and generator. Using the generator in a specific domain of application then involved classifying the categories of the domain ontology in terms of the categories provided by the Upper Model. The linguistic components of the generator were defined only to rely on distinctions made in the Upper Model.

The Upper Model organisation therefore takes on the task of classifying knowledge according to its possibilities for linguistic expression. This relationship is depicted graphically in Fig. 17.2. The Generalized Upper Model now takes this basic model further, refining and extending the categories maintained and, moreover, applying principles of ontological engineering to the categories proposed. Thus, while the original Upper Model, as with other linguistically motivated organisations such as WordNet, FrameNet, VerbNet etc., did not include explicit axiomatisations of its categories, current developments of the Generalized Upper Model seek also to find axiomatisations for the linguistically motivated categories proposed; there is also work in progress exploring the use of the Generalized Upper Model for automatic analysis, thereby extending beyond its previous use for language generation (cf. Bateman, 2010).

Fig. 17.2 Traditional relation between the generalized upper model and domain ontologies



The top-level categorizations found in the Generalized Upper Model follow the basic category distinctions broadly familiar from Jackendoff above and other linguistically motivated organisations: the semantic categories corresponding to entire clauses are termed **configurations**, while those corresponding to the phrases that can occur as subconstituents of clauses are termed **elements**. By virtue of the broader grammatical basis accepted for motivation, however, the model goes on to provide substantially more detail. Distinct subtypes of configurations are motivated according to general classes of clauses that behave differently within the grammar. Thus, the first-level distinctions drawn are: **Being&Having**, **Saying&Sensing**, and **Doing&Happening** – the grammatical clauses falling under these categories each have a distinctive range of grammatical phenomena that can be used for their identification. This is considered symptomatic of distinct underlying semantic categories and is therefore similar to VerbNet’s use of alternations, but goes further to include differences in valency patterns, tense usage, possible circumstantial information, selection restrictions and much more (cf. Halliday and Matthiesen, 2004).

Each of these categories also commits to a particular distinct set of semantic roles. This differs from VerbNet’s rather more traditional adoption of a generic set of semantic roles that potentially hold for any verb, and to a certain extent resembles FrameNet’s adoption of frame-specific roles for frames higher in the frame-inheritance hierarchy. Issues of semantic roles are not straightforward, however, and there have been few convincing attempts to provide sound axiomatisations for the distinct kinds of “participation” found. Some possible definitions from ontological perspectives are discussed by Sowa (1995) and Loebe (2003); such approaches tend to fall back within our model (I) above, however, and do not bring out the particularly linguistic contribution that such roles involve. Several other non-linguistically oriented ontologies include inventories of similar roles, such as SUMO’s **CaseRoles**, but these are freely used across all definitions, not only those that are concerned with linguistic semantics.

The Generalized Upper Model approach as a whole shows some similarities with the proposals of Cimiano and Reyle (2006), who argue that linguistic semantics should incorporate aspects of foundational ontologies. This they term *foundational semantics*, which

... is concerned with identifying that abstract meaning layer which remains constant across domains and applications. . . . From a theoretical point of view, foundational semantics aims at identifying the core components of the domain-independent meaning layer as well as to clarify their interplay, thus contributing to the understanding of the principles of semantic construction (Cimiano and Reyle, 2006: 52).

In distinction to the Upper Model, however, the starting point they adopt for their proposed account is given by established non-linguistic foundational ontologies, such as DOLCE (cf. [Chapter 13](#)). This is then only weakly connected to the concrete linguistic demands of the semantics of particular languages. Whereas it is to be hoped that such foundations will feed into improving the adequacy of linguistically motivated semantic organisations also, there remain many open questions about how these levels of description are best to be brought together. One possibility would be to embed a linguistically motivated organisation as a *description* within the Descriptions and Situations ontological extension proposed by Gangemi and Mika (2003), but the precise relationships between all these approaches still needs to be explored. Several approaches to this complex issue can be found in Huang et al. (2008).

17.4 Discussion

Given the range of, sometimes quite substantial, organisations of information that are being developed on the basis of linguistic evidence, including both lexical and grammatical motivations, it is natural that there is a considerable effort currently being extended to relate them. It remains to be seen how these efforts will continue and what the results will be.

For lexically-based organisations, such as WordNet and FrameNet, it can be expected that they will benefit substantially from a more rigorous foundation in ontology proper, although the extent to which the organisations drawn on lexical distinctions will match those motivated ontologically is still unclear. The utility of domain ontologies for natural language processing is also being increased by relating such ontologies to lexical resources: for example, there are now mappings between WordNet and many of the “middle level” extensions to SUMO. Connections between WordNet, VerbNet and FrameNet are also under development. Adding in the ability to move *across* WordNets for distinct languages, for example, by means of the *inter-lingual index* introduced by EuroWordNet, is also very promising. For grammatically-based organisations, such as VerbNet and GUM, their relationship with foundational ontologies still requires clarification. This work will move us further towards understanding the relationship between linguistic semantics and ontology, which becomes increasingly necessary as more sophisticated natural language processing is attempted (cf. Stone, 2003b).

We can also expect moves into further areas of useful interactions between linguistics, language processing and ontology that have not so far been prominent – for example, the increasingly explicit and broad coverage of speech acts (Bunt and Girard, 2005) and generalisations of linguistic information across different modalities (e.g. Niekrasz and Purver, 2006) are both being considered from an ontological perspective. Moreover, several dialogue systems already draw heavily on ontological characterisations of both their domains of discourse and their possibilities for action (e.g. van Diggelen et al., 2007).

We can also consider explicit representations of linguistic information itself, rather than of the semantic or ontological categories that languages may help

impose on the world. Such information includes details of types of clauses, parts of speech, morphological information and linguistic features of all kinds. Ontologies of this type are now being standardised in order to improve information exchange among different communities of linguistic researchers. The two most developed of these initiatives are GOLD (General Ontology for Linguistic Description: Farrar et al., 2002) and the Linguistic Annotation Framework (Ide et al., 2003; ISO/TC 37/SC 4). Such efforts can also be added to existing ontologies to further characterize any categories they may already have concerning language. For example, Farrar (2003) adds aspects of the GOLD ontology to SUMO, while Buitelaar et al. (2006) explore a combination of current linguistic standardisation efforts within a DOLCE/SUMO-hybrid called SWIntO (Oberle et al., 2007).

A driving motivation for all of these efforts currently is the promise that a thorough incorporation of ontologically-relevant information holds out for improving automatic text analysis, both with respect to its robustness (via better access to linguistic information) and to the depth of semantic processing possible (via better access to domain knowledge). Particularly in the context of the Semantic Web, such capabilities are now urgently required. We can be sure, therefore, that the interaction between ontology and natural language processing will continue to intensify.

References

- Asher, N. 1993. *Reference to abstract objects in discourse*. Dordrecht: Kluwer Academic Publishers.
- Baker, C.F., C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet project. In Proceedings of the ACL/COLING-98, 86–90. Montreal, Quebec.
- Baker, C.F., and J. Ruppenhofer. 2002. FrameNet's frames vs. Levin's verb classes. In eds. J. Larson, and M. Paster. Proceedings of the 28th Annual Meeting of the Berkeley Linguistics Society, 27–38.
- Bateman, J.A. 1990. Upper modeling: Organizing knowledge for natural language processing. In Proceedings of the 5th International Natural Language Generation Workshop, Pittsburgh, PA, 54–60. Organized by Kathleen R. McKeown (Columbia University), Johanna D. Moore (University of Pittsburgh) and Sergei Nirenburg (Carnegie Mellon University). Held 3–6 June 1990, Dawson, PA. <http://acl.ldc.upenn.edu/W/W90/W90-0108.pdf>.
- Bateman, J.A. 2004. The place of language within a foundational ontology. In eds. A.C. Varzi, and L. Vieu. Formal Ontology in Information Systems: Proceedings of the 3rd International Conference on Formal Ontology in Information Systems (FOIS-2004), 222–233. Amsterdam: IOS Press.
- Bateman, J.A. 2010. Language and space: A two-level semantic approach based on principles of ontological engineering. *International Journal of Speech Technology* 13(1):29–48.
- Bateman, J.A., R. Henschel, and F. Rinaldi. 1995a. Generalized upper model 2.0: Documentation, Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany. <http://purl.org/net/gum2>.
- Bateman, J.A., B. Magnini, and G. Fabris. 1995b. The generalized upper model knowledge base: Organization and use. In ed. N.J.I. Mars. Towards very large knowledge bases: Knowledge building and knowledge sharing. Amsterdam: IOS Press.
- Bateman, J., J. Hois, R. Ross, T. Tenbrink, and S. Farrar. 2008. The generalized upper model 3.0: Documentation, SFB/TR8 internal report. Germany: Collaborative Research Center for Spatial Cognition, University of Bremen.

- Bierwisch, M., and E. Lang, eds. 1989. *Dimensional adjectives. Grammatical structure and conceptual interpretation*. Berlin: Springer.
- Buitelaar, P., T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel, and P. Cimiano. 2006. LingInfo: Design and applications of a model for the integration of linguistic information in ontologies. In Proceedings of the OntoLexWorkshop at LREC, 28–32.
- Bunt, H., and Y. Girard. 2005. Designing an open, multidimensional dialogue act taxonomy. In Proceedings of DIALOR 05', Nancy, 37–44. <http://dialor05.loria.fr/Papers/05-harryann.pdf>
- Burchardt, A., A. Frank, and M. Pinkal. 2005. Building text meaning representations from contextually related frames – A case study. In Proceedings of the 6th International Workshop on Computational Semantics, IWCS-6', Tilburg, The Netherlands.
- Cimiano, P., and U. Reyle. 2006. Towards foundational semantics – Ontological semantics revisited. In Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS). IOS Press. <http://www.aifb.uni-karlsruhe.de/WBS/pci/Publications/fois06.pdf>
- Cruse, D.A. 1986. *Lexical semantics*. Cambridge, MA: Cambridge University Press.
- Dahlgren, K. 1995. A linguistic ontology. *International Journal of Human-Computer Studies* 43(5/6):809–818.
- Dale, R., H. Moisl, and H. Somers, eds. 2000. *A handbook of natural language processing: Techniques and applications for the processing of language as text*. New York, NY: Marcel Dekker.
- Davidson, D. 1967. The logical form of action sentences. In *The logic of decision and action*, ed. N. Rescher, 81–95. Pittsburgh, PA: University of Pittsburgh Press.
- Doran, C., D. Egedi, B.A. Hockey, B. Srinivas, and M. Zaidel. 1994. XTAG system – A wide coverage grammar for English. In Proceedings of the 15th International Conference on Computational Linguistics (COLING 94), Kyoto, Japan, vol. II, 922–928.
- Farrar, S. 2003. An ontology for linguistics: Extending SUMO with GOLD. In Proceedings of the 2003 IEEE International Conference on Natural Language Processing and Knowledge Engineering, IEEE, Beijing, PRC.
- Farrar, S., W.D. Lewis, and D.T. Langendoen. 2002. A common ontology for linguistic concepts. In Proceedings of the Knowledge Technologies Conference, Seattle.
- Fillmore, C., and B. Atkins. 1998. FrameNet and lexicographic relevance. In Proceedings of the International Conference on Language Resources and Evaluation (LREC), Granada, Spain.
- Fillmore, C.J. 1976. Frame semantics and the nature of language. In *Origins and evolution of language and speech*, ed. S. Harnad, 155–202. New York, NY: Academy of Sciences.
- Gangemi, A., N. Guarino, A. Oltramari, and S. Borgo. 2002. Cleaning-up WordNet's top-level. In Proceedings of the 1st International WordNet Conference.
- Gangemi, A., and P. Mika. 2003. Understanding the semantic web through descriptions and situations. In Proceedings of ODBASE 2003.
- Guarino, N. 1998. Some ontological principles for designing upper level lexical resources. In eds. A. Rubio, N. Gallardo, R. Castro, and A. Tejada. Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC), Granada, Spain, 527–534.
- Halliday, M.A.K., and C.M.I.M. Matthiessen. 1999. *Construing experience through meaning: A language-based approach to cognition*. London: Cassell.
- Halliday, M.A.K., and C.M.I.M. Matthiessen. 2004. *An introduction to functional grammar*, 3rd edition. London: Edward Arnold.
- Huang, C., N. Calzolari, A. Gangemi, A. Lenci, A. Oltramari, and L. Prevot, eds. 2008. *Ontologies and the lexicon*. Cambridge, MA: Cambridge University Press.
- Ide, N., L. Romary, and E. de la Clergerie. 2003. International standard for a linguistic annotation framework. In Proceedings of the HLT-NAACL'03 Workshop on the Software Engineering and Architecture of Language Technology.
- Jackendoff, R. 1983. *Semantics and cognition*. Cambridge, MA: The MIT Press

- Kipper-Schuler, K. 2005. VerbNet: A broad-coverage, comprehensive verb lexicon, PhD thesis, Philadelphia, PA: Computer and Information Science Department, University of Pennsylvania. <http://repository.upenn.edu/dissertations/AAI3179808/>.
- Kuhn, W. 2001. Ontologies in support of activities in geographical space. *International Journal of Geographical Information Science* 15(7):613–631.
- Langacker, R.W. 1987. *Foundations in cognitive grammar. Vol. 1. Theoretical prerequisites*. Stanford, CA: Stanford University Press.
- Levin, B. 1993. *English verb classes and alternations: A preliminary investigation*. Chicago/London: University of Chicago Press.
- Loebe, F. 2003. An analysis of roles: Towards ontology-based modelling, OntoMed Report 6, Institute for Medical Informatics, Statistics and Epidemiology (IMISE). Germany: University of Leipzig. <http://www.onto-med.de>.
- Mann, W.C., Y. Arens, C.M.I.M. Matthiessen, S. Naberschnig and N.K. Sondheimer. 1985. Janus abstraction structure – draft 2. Technical report, Marina del Rey, CA: USC/Information Sciences Institute. (Circulated in draft form only.)
- Minsky, M. 1975. A framework for representing knowledge. In *The psychology of computer vision*, ed. P.H. Winston, 211–277. New York, NY: McGraw-Hill. Reprinted in eds. R.J. Brachman, and H.J. Levesque. 1985. *Readings in knowledge representation*, Los Altos, CA: Kaufman.
- Niekrasz, J., and M. Purver. 2006. A multimodal discourse ontology for meeting understanding. In eds. S. Renals, and S. Bengio. Proceedings of MLMI 2005, LNCS 3869, 162–173. Berlin/Heidelberg: Springer.
- Niles, I., and A. Pease. 2003. Linking lexicons and ontologies: Mapping wordnet to the suggested upper merged ontology. In Proceedings of the IEEE International Conference on Information and Knowledge Engineering, IEEE, 412–416.
- Nirenburg, S., and V. Raskin. 2001. Ontological semantics, formal ontology, and ambiguity. In eds. C. Welty, and B. Smith. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, 151–161. New York, NY: ACM Press.
- Oberle, D., A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougouie, S. Baumann, S. Vembu, M. Romanelli, P. Buitelaar, R. Engel, D. Sonntag, N. Reithinger, B. Loos, et al. 2007. DOLCE ergo SUMO: On foundational and domain models in the SmartWeb integrated ontology (SWIntO). *Journal of Web Semantics* 5:156–174.
- Park, B. 1995. A language for ontologies based on objects and events. In Proceedings of the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Menlo Park, CA: AAAI Press.
- Parsons, T. 1990. *Events in the semantics of english: A study in subatomic semantics*. Cambridge, MA/London: MIT Press.
- Pease, A., and I. Niles. 2002. Practical semiotics: A formal theory. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE '02), Las Vegas, NV.
- Sowa, J.F. 1995. Top-level ontological categories. *International Journal of Human-Computer Studies* 43(5/6):669–686.
- Stone, M. 2003a. Knowledge representation for language engineering. In *A handbook for language engineers*, CSLI Lecture Notes 164, ed. A. Faghaly, 299–366. Stanford, CA: CSLI Publications.
- Stone, M. 2003b. Ontology and description in computational semantics. In Proceedings of the 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems at the 18th IJCAI. <http://www.ida.liu.se/labs/nlplab/ijcai-ws-03/papers/stone.pdf>.
- Talmy, L. 1987. Figure and ground in complex sentences. In *Universals of human language*, ed. J.H. Greenberg, 625–649. Stanford, CA: University of California Press.
- Talmy, L. 2000. *Towards a cognitive semantics*. A Bradford Book. Cambridge, MA: MIT Press.
- Talmy, L. 2006. The fundamental system of spatial schemas in language. In ed. B. Hampe, *From perception to meaning: Image schemas in cognitive linguistics*, 37–47. Berlin: Mouton de Gruyter

- Trautwein, M., and P. Grenon. 2004. Roles: one dead armadillo on WordNet's speedway to ontology. In Proceedings of the 2nd Global WordNet Conference, 341–346. Brno, Czech Republic. <http://www.fi.muni.cz/gwc2004/proc/118.pdf>.
- van Deemter, K., and S. Peters, eds. 1996. *Semantic ambiguity and underspecification*. Stanford, CA: CSLI.
- van Diggelen, J., R. Beun, F. Dignum, R. van Eijk, and J.-J. Meyer. 2007. Ontology negotiation: Goals, requirements and implementation. *International Journal of Agent-Oriented Software Engineering* 1(1):63–90.
- Vendler, Z. 1957. Verbs and times. *Philosophical Review* 66:143–160.
- Vossen, P., P. Diez-Orzas, and W. Peters. 1997. The multilingual design of EuroWordNet. In Proceedings of the ACL/EACL-97 Workshop on Automatic Information Extraction and Building Lexical Semantic Resources for NLP applications, Madrid: Association for Computational Linguistics.
- Wierzbicka, A. 1988. *The semantics of grammar*, Studies in Language Companion Series 18, Amsterdam/ Philadelphia, PA: John Benjamins Publishing Company.
- Zarri, G.P. 2005. An n-ary language for representing narrative information on the web. In eds. P. Bouquet, and G. Tummarello. Proceedings of SWAP 2005 – Semantic Web Applications and Perspectives, Proceedings of the 2nd Italian Semantic Web Workshop, Trento, Italy: University of Trento, 14–16 December 2005, vol. 166 of CEUR Workshop Proceedings, CEUR-WS.org. <http://www.ceur-ws.org/Vol-166/63.pdf>.

Chapter 18

Business Ontologies

Peter Rittgen

18.1 Introduction

Ontologies are typically divided into foundational (or top-level), domain and application ontologies (Bugaitė and Vasilecas, 2005). Foundational ontologies cover the most general concepts that can be expected to be common to all domains, such as “individuals” vs. “universals” or “substantials” vs. “moments”. They are therefore domain-independent. Domain ontologies are tailored for a specific area of human activity, e.g. medicine, electrical engineering, biology or business. Application ontologies further restrict attention to a particular activity in a domain, e.g. the diagnosis of lung diseases in medicine or a computer-based order handling system in business. Figure 18.1 shows the level architecture and names a few examples on each level.

It can be argued, though, whether three levels of ontology are adequate to cover the whole breadth of ontological endeavors. In the business domain, for example, we can identify a number of dimensions that justify further ontological levels. Let us consider a few examples. We distinguish between private-sector and public-sector organizations. Each organization belongs to some industry (banking, car manufacturing, retail, etc.) and it is divided into functional units such as procurement, production, marketing, sales and so on. Along the hierarchy we have the strategic, tactical and operational levels. In addition to these we might also consider a level below the application domain level, the personal level that takes into account, e.g. the way in which an individual uses a particular information system for a particular task which is often different from the way others use the same system for the same or a similar task (Carmichael et al., 2004; Dieng and Hug, 1998; Haase et al., 2005; Huhns and Stephens, 1999).

P. Rittgen (✉)
Vlerick Leuven Gent Management School, Leuven, Belgium; University of Borås,
Borås, Sweden
e-mail: Peter.Rittgen@vlerick.com

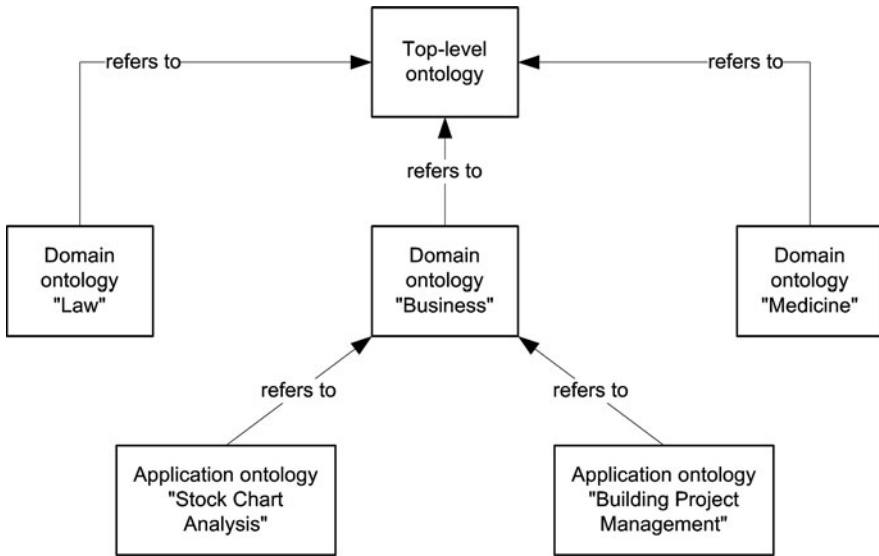


Fig. 18.1 Ontology levels

18.1.1 Domain-Level Ontologies

The diversity of phenomena along all these dimensions makes it difficult to find an adequate level of abstraction that fits the whole business domain. In organizational theory a number of metaphors have been suggested to understand and explain organizational behavior at a high level of abstraction. Metaphors establish a link between a source field and a target field and explain phenomena in the target field in terms of the source field. For organizational theory as a target field the following source fields have been proposed: the machine metaphor (Scott, 1997), living systems (biology) (Kendall and Kendall, 1993), open systems (Flood, 2005), the brain metaphor (Gareth, 1997), learning systems (Senge, 1990), social networks (Davern, 1997), complex adaptive systems (Anderson, 1999), autopoietic social systems (Luhmann, 1990) and so on. A metaphor is a vehicle for explaining a target domain in terms of a source domain, e.g. the steering of a ship (source) as a metaphor for the steering of a company (target). Using a metaphor therefore implies a shift of domain. Existing ontologies for the source domain can therefore be transferred to the business domain.

But metaphors also imply some severe restrictions. By viewing organizations as, e.g. living systems we fail to capture those parts of organizational behavior that are not found in biology. Established approaches to a business ontology draw therefore on a number of different related theories to develop a richer picture of the domain. Theoretical contributions can come from communication theories, e.g. Speech Act Theory (Austin, 1975; Searle, 1997, 1999) and Theory of Communicative Action (Habermas, 1984); social theories, e.g. Actor Network

Theory (Law, 1992; Walsham, 1997) or Structuration Theory (Giddens, 1986); economic theories, e.g. Agency Theory (Jensen and Meckling, 1976; Ross, 1973) or Transaction Cost Economics (Klein et al., 1978; Williamson, 1981, 1983, 1998) and others.

Examples of existing approaches to a general ontology of the business domain are: Core Enterprise Ontology (CEO) (Bertolazzi et al., 2001), Edinburgh Enterprise Ontology (EEO) (Uschold, King, Moralee and Zorgios, 1998), Toronto Virtual Enterprise (TOVE) (Fox and Gruninger, 1998), Business Model Ontology (BMO) (Osterwalder, 2004), e³value Ontology (EVO) (Gordijn, 2004), Socio-Instrumental Pragmatism (SIP) (Goldkuhl, 2002, 2005) and Enterprise Ontology (EO) (Dietz, 2006).

CEO introduces a framework that only specifies the concepts that are common to the whole business domain. They are grouped into four areas: Passive entities (business objects); active entities (actors, agents); transformations (actions, processes); and conditionals (business goals and rules, constraints and states). It is up to the ontology designer to build the actual ontology refining the basic concepts into the specific ones of the respective application domain.

EEO takes a different approach. Instead of just providing a framework they actually specify a supposedly complete repository of detailed enterprise terms. The ontology user therefore only has to choose the ones s/he needs for the particular application. EEO provides both a natural-language and semi-formal definition for all the terms ranging from the foundational concepts of the meta-ontology (e.g. entity, relationship, actors) to the domain concepts that are divided into 4 main areas: activities, organization, strategy and marketing.

TOVE is similar to EEO in that they also try to capture a complete enterprise terminology in a number of ontologies. The base ontology is a general Activity ontology from which more specific ontologies are derived such as Organization Ontology and Resource Ontology. But contrary to EEO the concepts and relations are defined in a completely formal way including theorems and proofs of important properties such as soundness and completeness.

BMO is based on the balanced scorecard and defines four so-called pillars: Product, customer interface, infrastructure management and financial aspects. These pillars are then refined into a set of nine building blocks: Value proposition (concerning the product); target customer, distribution channel and relationship (concerning the customer interface); value configuration, capability and partnership (infrastructure management); and cost structure and revenue model (financial aspects). The important difference to the above mentioned approaches is that BMO takes a wider view of a business including also concepts that lie outside the focus of the actual enterprise but which have considerable impact on it such as the ones related to the customer interface. BMO also introduces the concept of value (of a product or service) whereas other business ontologies are often restricted to the cost perspective.

After having discussed a few business ontologies on a general level we shall describe two examples in greater detail. They are: Socio-Instrumental Pragmatism (Goldkuhl, 2002, 2005) and Enterprise Ontology (Dietz, 2006). They represent two

diagonally opposed ends of the spectrum spanned by the dimensions scope and degree of elaboration. This allows the reader to get an impression of the bandwidth of the approaches. The former has a wide scope that covers any kind of social behavior including business action. But it is not yet a full-blown ontology but rather a basic taxonomy complemented by a set of relations between the basic concepts. Enterprise Ontology, on the other hand, is exactly the opposite: It is a highly elaborated and formalized ontology that provides a substantial level of detail regarding concepts and their relations, and also a set of axioms defining the semantics. But its scope is much narrower as it makes a clear commitment to a very specific conceptualization of the business world, thereby excluding other points of view. Enterprise Ontology has been criticized for that (Goldkuhl and Lind, 2004; Verharen, 1997) but due to its rigidity this approach can nevertheless serve as an illustrative example of a business ontology. The chosen example ontologies are described in the sections Socio-Instrumental Pragmatism and Enterprise Ontology, respectively.

18.1.2 Application-Level Ontologies

As such a general ontology of the business domain cannot be used directly in any concrete business application. It is therefore necessary to have at least one more level, the application ontology. Some researchers suggest additional levels, e.g. task ontologies (Guarino, 1998). But instead of introducing a multitude of levels we propose to interpret them as different domain ontologies instead because most of the interesting problems already occur in the case of a second level. So we only abstract from the complexity levels that do not contribute to our discussion. We do not argue that a reduction to three levels is indeed sufficient. According to this definition, a domain ontology can be task-specific, company-specific etc. We illustrate most of the following discussions in the context of information systems where most of the business ontology research is located.

When we take a look at the application-ontology level we discover that the idea of having a separate ontology for every application is fraught with a severe problem as many individuals and organizations make use of several applications within the context of a single task or business process. Let us consider two of the solutions that have been proposed to solve this problem. The first one, a bottom-up approach, aims at integrating the affected application ontologies, each of which could have been developed independently, to derive a higher-level domain ontology for the task or the specific organization. An example of this is given in (Corbett, 2003).

The second solution is top-down. It assumes the existence of a library of ontologies that is used to build an application ontology (e.g. on the task level) by re-using existing domain ontologies (e.g. on the business process level). Systems that support this are called ontology library systems. Examples of such systems are WebOnto (Domingue, 1998), Ontolingua (Farquhar et al., 1997) and SHOE (Heflin and Hendler, 2000).

18.2 Socio-Instrumental Pragmatism

Socio-Instrumental Pragmatism (SIP) is an ontology that combines social, communicative and instrumental aspects of business behavior. The basic assumption behind SIP is that business action is performed by a human being (possibly on behalf of an organization) with the help of some instrument (tool) to create a result for another human being or organization. The social aspect consists in the action being directed from one human being to another. The instrumental aspect aims at the technical (or material) dimension of this action. The instrument can be anything from an axe to a complex IT (Information Technology) system. As social action requires communication between individuals the communicative element is also present.

SIP is based on six ontological concepts (Goldkuhl, 2002): Human being, human inner world, human action, sign, artifact and natural environment. They are defined in the following paragraphs.

Human being is the most fundamental concept because they are the actors in the social world described; they act in the world based on meanings and perceptions that they derive from the world.

The *human inner world* represents the knowledge that a human being has acquired over time about themselves and the external world; this inner world is intended to be seen as part of the human being.

Human action is an important aspect of the human being; it can be overt, which means that the actions are intended to intervene in the external world, thus trying to change something about it. And they can be covert, when they are aimed to change some other human being's inner world; covert actions try to change knowledge that resides in the human inner world.

Signs are the result of communicative actions; for instance, when we write a note saying, "I will be at the store", the writing of the note is by itself a communicative action, but the note created is a sign which will mean something to the person who will read it.

Artifacts are things which are not symbolic and not natural but which are material and artificially created. Examples of artifacts are cars, clothes, a knife, etc. The difference between signs and artifacts is that while signs are intended to mean something to someone (symbolic), artifacts perform material actions. For instance, a human might use a knife (artifact) to cut some carrots, i.e. artifacts are needed to perform material actions.

Natural environment means the objects present in the environment that are not artificially created by humans (e.g. trees).

Figure 18.2 shows the different realms of the world according to the SIP ontology.

18.2.1 Restructuring the Taxonomy

In an attempt to formalize the conceptual system of SIP and the inter-concept relations, we have developed a meta-model in the form of a class diagram of the UML

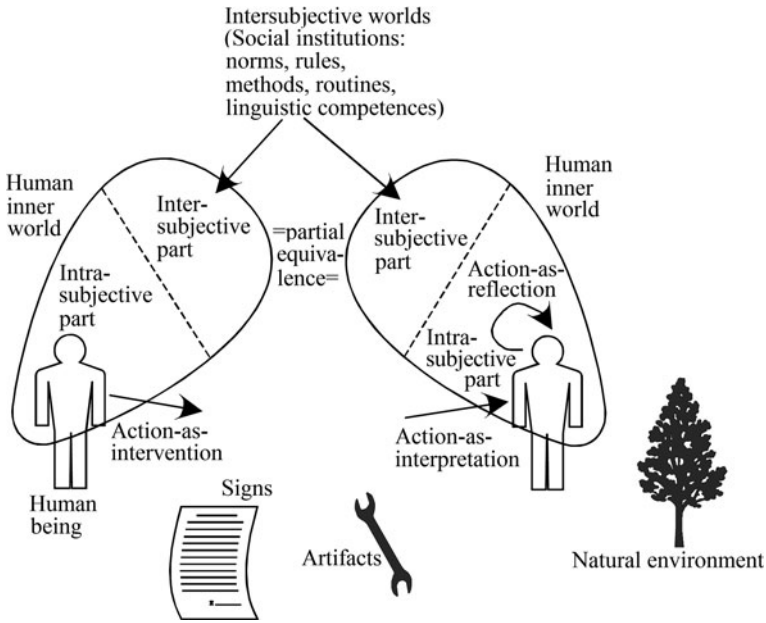


Fig. 18.2 Realms of the world in SIP (Goldkuhl, 2002)

(Unified Modeling Language). A meta-model defines a language for modeling, i.e. it specifies the terms of that language and the way in which they are related and thereby indirectly also defines the semantics to some degree by setting constraints that restrict the use of the language to the admissible models and excluding the ones that are not meaningful. Meta-models can be seen as one way of representing an ontology where the terms of the modeling language are the concepts of the ontology.

A UML Class Diagram is a language that allows for the expression of meta-models (as well as object-level models of a domain). It offers classes, associations and generalization as meta-concepts (among others). Classes are collections of objects that share common attributes and behavior and they can be used to represent concepts of an ontology. Associations are relations between classes which refer to the concept relations in the ontology. Generalization declares one class as the generalization of another thereby introducing a taxonomical relation in terms of the ontology. For further details about UML see (OMG, 2005, 2006).

The meta-modeling process involved a restructuring of the taxonomy suggested by (Goldkuhl, 2002). Human action has been generalized to (social) action and human being to actor. The other concepts become sub-concepts of object. This leaves us with four main concepts: Actors, agents, objects and actions, where agent is a new concept that was introduced to handle artifacts that can perform actions.

18.2.1.1 Actors

Actors are the main entities in our model, and they can perform either as locutor or addressee within the communicative context. When actors perform actions that are directed towards another actor we speak of social actions. They can be performed either in a human-human relation or in a human-artifact-human relation. When performing as locutor the actor is trying to change some aspect of the world by means of his/her actions. For instance, when a person pays the phone bill she is trying to avoid the interruption of her phone service. When performing as addressee the actor receives and interprets an action directed to him and can act himself as a consequence of that action. Taking our example the addressee will be the phone company, which at the moment of receiving the payment will not make any attempt to interrupt the customer's phone service.

Besides locutor and addressee we can distinguish between organizational actors and human actors. The former is an actor that performs as an agent on behalf of the organization; the latter performs an action on behalf of herself.

18.2.1.2 Objects

An object may be physical or conceptual and it may be formed by other objects or related to them, but every object is unique (Embley et al., 1992). Under the object concept we have artificial and natural objects. Artificial are those that are created by human beings, natural objects are those created by nature and found in the environment. Among the artificial objects we have artifacts (material objects) and signs (can be material or immaterial). Artifacts are created to extend actors' capabilities. An artifact is seen as a tool. Signs on the other hand are not tools but messages in a static phase waiting to be interpreted by actors or artifacts. A message can take either a physical form (a written text) or a non-physical form (an utterance) (Goldkuhl, 2002).

We can distinguish between 4 different types of artifacts: static, dynamic, automated and multi-level. Static artifacts are those that cannot perform any operation by themselves, e.g. a stone, a knife or an axe. Dynamic objects are those capable of performing some operations by themselves but they need constant control by a human being to function properly, for example a car or a driller. Automated artifacts are those that can operate entirely by themselves and only need to be started by an actor. Here we can mention a washing machine as an example (Goldkuhl and Ågerfalk, 2005).

Multi-level artifacts are those that have a mix of capabilities and can perform either as static, dynamic or automated artifacts depending on the circumstances. Multi-level artifacts have an important property which is the capability of creating and interpreting signs. They lack consciousness and are ruled by a pre-defined set of instructions that serve as a guide to perform the pre-defined actions they do. IT systems are an example of multi-level artifacts. Signs can be created either by human beings or artifacts, and every sign can be interpreted by human beings only,

by artifacts only, or by both (Goldkuhl, 2005). A written note is a sign; an utterance performed by an actor is another example of a sign as well as a ticket printed by a system in an electronic store.

18.2.1.3 Actions

The objective of human actions is to change something in the world. They can be communicative or material. The main difference between these two types of actions lies in the fact that communicative actions are intended to change knowledge. Knowledge is implicitly meaningful to someone; and knowledge handling is an exclusive characteristic of actors within an IS (Information System). On the other hand, material actions are aimed at material conditions and aspects of the world which are meaningful to someone. They are intended to change something physical in the external world. As a human characteristic, knowledge can be learned through actions, either communicative actions (for instance, a conversation) or material actions (e.g. when studying an object). Knowledge is the result of the actor's interpretation of both communicative and material actions, and it can be acquired in a social context (from other actors transferring knowledge e.g. in a classroom) or in a non-social context (a person reading a book on her own) (Goldkuhl, 2001).

We can divide actions into i-actions (intervening actions) and r-actions (receiving actions). I-actions are those intended to make a change in the external world, e.g. the action of opening a window is intended to change a particular aspect of the external world (the window will move from closed to open). R-actions are those executed covertly, for example when two people are going out and person A tells B "It's cold outside" (communicative i-action). Then person B listens and interprets the message (r-action). Following this, person B will take a jacket on the way out (material i-action) (Goldkuhl, 2001). Among i-actions and r-actions we have indefinite and pre-defined actions.

Indefinite actions are those performed by humans. We call them indefinite since it is not certain how they will be performed by the actor. The same action can vary from actor to actor. When two employees are ordered to clean a shelf, they will both do it but not in the same way, one can do it better or faster than the other one. Indefinite actions can be either r-actions or i-actions. On the other hand we have pre-defined actions which are performed by artifacts. These actions will always be performed in the same way following previously programmed instructions (Goldkuhl, 2005). Pre-defined actions are i-actions, since they are intended to change an aspect of the external world. Among indefinite and pre-defined actions we find both communicative and material actions.

Both material and communicative actions aim at changing an aspect of the world surrounding the actor or artifact, but communicative actions have at least two phases where actor A first performs a communicative action that is directed towards another actor or artifact B. In the second phase B (if A was successful) executes the action that A desired. Although material, the last action can sometimes be performed without an initial communicative action.

Material and communicative actions within organizations form patterns. Although human beings perform them, we can also say that the organization acts. An

organizational action has human origins and purposes and is done through humans, by humans or by artifacts that act on behalf of the organization (Goldkuhl et al., 2001). We will consider actions as organizational if they constitute an interaction between two or more actors or agents of the organization within an organizational context. We can say that a worker at a clothes factory using a sewing machine to manufacture clothes is performing an organizational action. He is acting to perform an organizational objective (to produce clothes). But for instance a man on a farm that goes to the forest to chop wood using an axe, although using an artifact to perform the action, is not performing an organizational action since there is no organizational purpose if he merely burns the wood to warm up his house.

When performing actions by means or with the help of IT systems we can distinguish between three different types of actions: interactive, automatic and consequential actions. Interactive actions are supported by and performed through IS and they consist of one or more elementary interactions. Elementary interactions (e-actions) consist of 3 phases: a user action, an IT system action and a user interpretation (Goldkuhl, 2001). Let us take the example of an online bank transfer done by the user online. The user will initially introduce his username and password to access the bank system (phase 1), after this the IT system will check in the database if the information is correct and if it is it will grant access to the user and display a welcome screen (phase 2). The welcome screen is interpreted, and the user now knows that he can start his transaction. This is the end of the elementary interaction. Later on the user inputs the data to make the bank transfer, such as account number, amount to be transferred, etc. (phase 1 of a second e-interaction), and so on.

Automatic actions are performed by IT systems that produce messages for the actors or other systems. They are done entirely without human intervention. Let us take the banking system again: After logging on, a message pops up telling the customer that the due date for the credit card payment is very close. The system will execute this operation by itself and present it to the user.

Consequential actions are those performed as a consequence of a message. Taking the bank example again, when the customer sees that his payment is due he might proceed to execute the payment, or he might decide not to do it and wait for the final day.

18.2.1.4 Agents

Agents are a special type of object. They are created by actors, and perform actions to help them complete their tasks. They can be seen as servants of actors, but they have a level of communicative capabilities that allow them to act as communicative mediators, and they are also capable of creating signs for the actors or other agents to interpret. The difference between agents and actors lies in the fact that actors can perform social actions while agents cannot (Rose and Jones, 2004).

IT systems can perform as agents. They can be seen as static artifacts, automated artifacts or dynamic artifacts (Goldkuhl and Ågerfalk, 2005). In all three cases the common denominator is communication. Communication is seen as a kind of action that IT systems can perform and by doing so they become communication mediators. IT systems as well as actors have the capability to create signs and

to process them. Actors can also interpret them (Goldkuhl, 2001). The relation between the signs and their interpreters/processors is called pragmatics. Within IS pragmatics, actions are divided into those that occur within the sign transfer and consequential actions that are performed in response to the transferred sign (Goldkuhl and Ågerfalk, 2000).

18.2.2 The Resulting Meta-model

As a result of the contemplations in the previous sections we have developed a meta-model (see Fig. 18.3) that covers the most important aspects of Socio-Instrumental Pragmatism as outlined in Section 18.2. Technically the meta-model takes the form of a UML class diagram with generalization/specialization and association.

18.3 Enterprise Ontology

Enterprise Ontology has its roots in the ontological frameworks of Bunge and Wittgenstein (Bunge, 1977; Wittgenstein, 2001). According to it a world can be in any of a number of states. The ability to move from one state to another is called transition. An occurrence of a transition is an event. An event is caused by an act. Worlds consist of two kinds of objects, *stata* and *facta*. A *statum* is a thing that exists in all states of the world independent of any act. A *factum* is the result of an act that brings it about. *Stata* can be declared (i.e. original) or derived (from other *stata*). Existence laws determine the possible states of the world by specifying which combinations of *stata* are allowed in a state (the state space).

A *factum* is associated with an event (i.e. the occurrence of a transition). Occurrence laws determine the possible orders of transitions by specifying which sequences of transitions are allowed (the transition space). An ontology is defined by specifying the state and transition spaces. It is done with the help of a language, the World Ontology Specification Language. It uses a graphical notation borrowed from conceptual modeling, i.e. ORM (Halpin, 2001).

18.3.1 World Ontology Specification Language

A *statum* can be declared intensionally via its properties, or extensionally by enumerating the instances that belong to the *statum* type. The intension is an n -ary predicate that is denoted as a rectangle with n horizontal fields with an inscribed placeholder each. Underneath we write a predicative sentence detailing the role of each placeholder. On top we write the name of the *statum* type in bold lower-case letters. The extension is a set denoted by a rounded rectangle and inscribed with the name of the *statum* type in capital letters. Intensionally defined *statum* types can be entified (extensionalized) by surrounding them with a rounded rectangle.

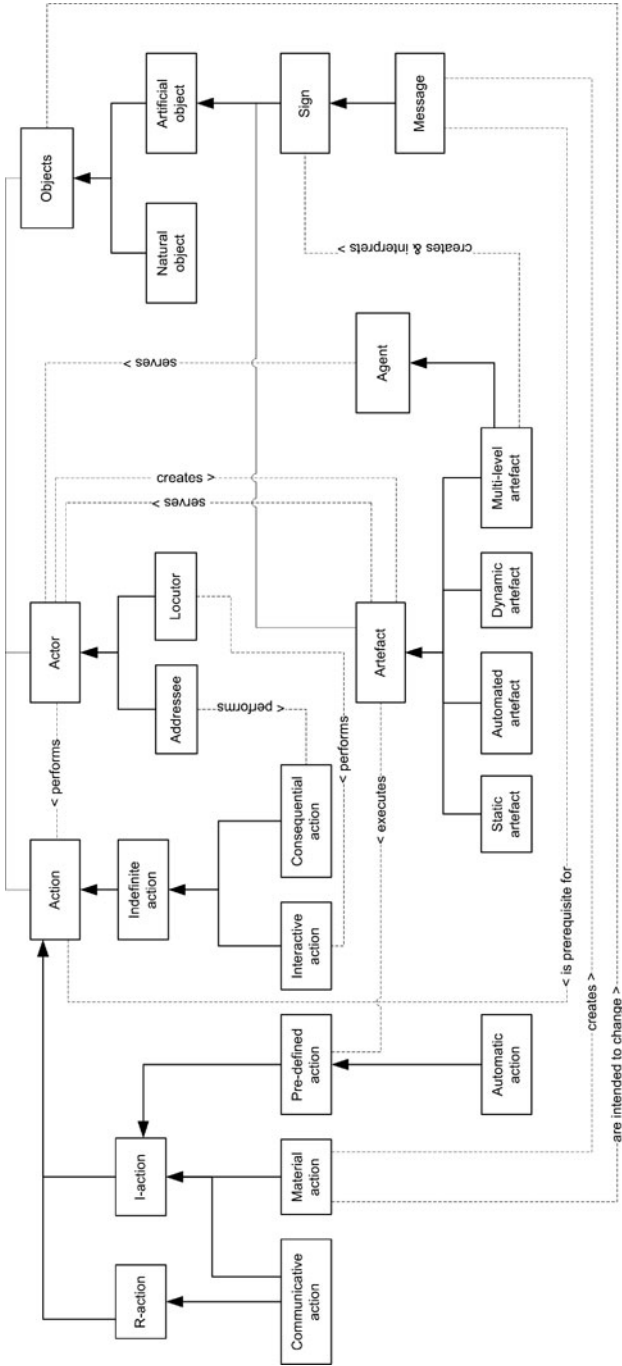


Fig. 18.3 Meta-model of SIP

	Category	Class	
Extensional			
Intensional	 X is a person	 X is a teacher	 X is the boss of Y

Fig. 18.4 Declaration of statum types

Categories are primal statum types in the sense of ultimate sortals according to (Guizzardi et al., 2004), i.e. they do not rely on other stata for their definition. They are denoted by a thick borderline around the respective shape. Classes are defined by establishing a reference to another class or a category. They are surrounded by a normal line. Figure 18.4 shows the different ways of declaring statum types.

We have 4 different types of existence laws that can be applied to restrict the space of allowed states: Reference laws, dependency laws, unicity laws and exclusion laws (see Fig. 18.5). With the help of a reference law we can define the domain of a placeholder in a statum type. In the example a lecturer must always be a teacher (but not every teacher is a lecturer). Dependency laws imply a mutual reference, e.g. every employee has a boss and everyone with a boss is an employee. A unicity law excludes that a statum can participate in multiple instances of the same predicate, for example, an employee cannot have two bosses. An exclusion law prevents a statum that is engaged in one relation to participate in another, e.g. a CEO does not have

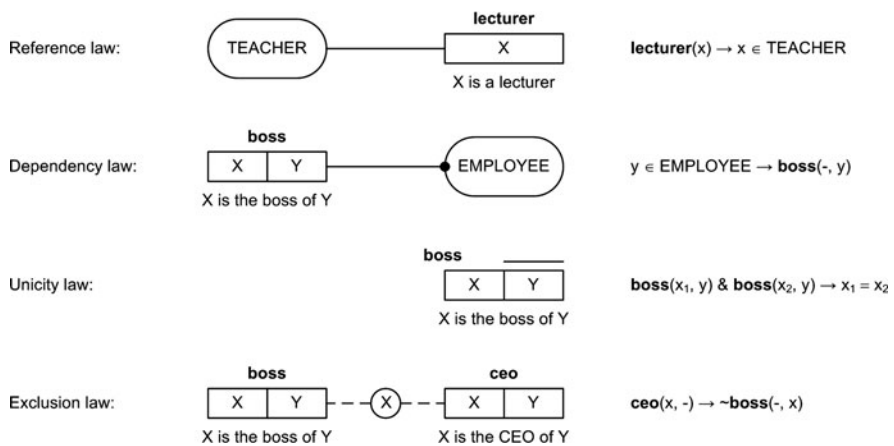


Fig. 18.5 Existence laws for stata

a boss as denoted by the exclusion relation between the boss and CEO objects in Fig. 18.5 (an encircled symbol *X* on a dashed line).

Statum types can also be derived from others. The shape of the derived type is in that case hatched (i.e. the shape's background area is filled with diagonal lines). Derivation rules include aggregation, generalization and specialization. Aggregation is the same as entification. Specialization and generalization are not complementary. For instance, *student* can be seen as a specialization of the category PERSON, but PERSON is not a generalization of *student*, *teacher* and other classes as PERSON is a primal type. The class VEHICLE, on the other hand, is a generalization of HORSE CART, BICYCLE, CAR, and so on. The latter are not specializations of the former as the first mode of transport that was developed was a horse cart and not a vehicle. Only later, when additional transport modes had been devised, did it make sense to generalize on their common ability to transport human beings and to address them collectively as vehicles.

Factum types are denoted by a diamond inscribed with a placeholder. Above it a predicative sentence specifies the transition that marks the occurrence of the factum (see Fig. 18.6, above the line). Occurrence laws indicate the restrictions on the transition space. They can prescribe a certain sequence, e.g. an order must have been placed before it can be shipped, or forbid it, e.g. an order that has already been shipped cannot be cancelled anymore (see Fig. 18.6, below the line).

18.3.2 The Axioms of Enterprise Ontology

The World Ontology Specification Language allows for the specification of almost any world (or domain). In order for the resulting ontology to be a business ontology a number of axioms must be satisfied. They are the operation axiom, transaction axiom, composition axiom and distinction axiom.

18.3.2.1 The Operation Axiom

The operation axiom claims that business action takes place in two worlds: the coordination world (or C-world) and the production world (or P-world). P-acts can be material (transport, manufacturing, storage) or immaterial (e.g. decisions). C-acts

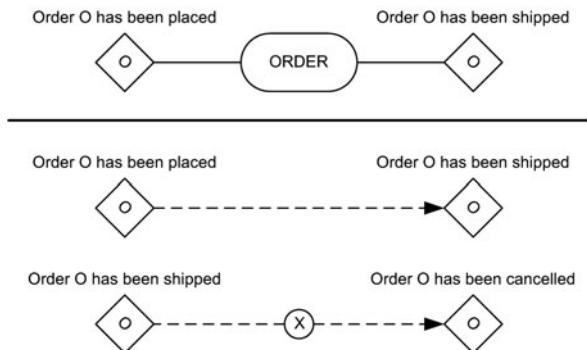
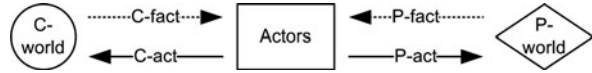


Fig. 18.6 Declaration of facts and occurrence laws

Fig. 18.7 Worlds of enterprise ontology



are used to coordinate, i.e. manage, the productive behavior. C-acts are communicative acts that allow us to establish and follow up commitments. Both types of acts lead to respective facts that in turn are evaluated by the actors to determine further acts. Actors in roles are minimal units of authority and responsibility. Figure 18.7 depicts the implications of the operation axiom.

18.3.2.2 The Transaction Axiom

The transaction axiom states that coordination acts are performed according to patterns. The standard pattern assumes that both parties go on confirming each other's acts. It is shown in Fig. 18.8. If the parties disagree they move to a different layer, the discussion layer to resolve the conflict. If this fails they can move to the discourse layer where general norms and values can be addressed. There are also additional patterns that deal with cancellation (Dietz, 2006).

Here we will only look at the standard pattern. In it two parties, called initiator and executor, try to establish and follow up a commitment that concerns the execution of a P-act by the executor. The transaction is divided into three phases, order, execution and result. The order phase is an actagenic (i.e. action generating)

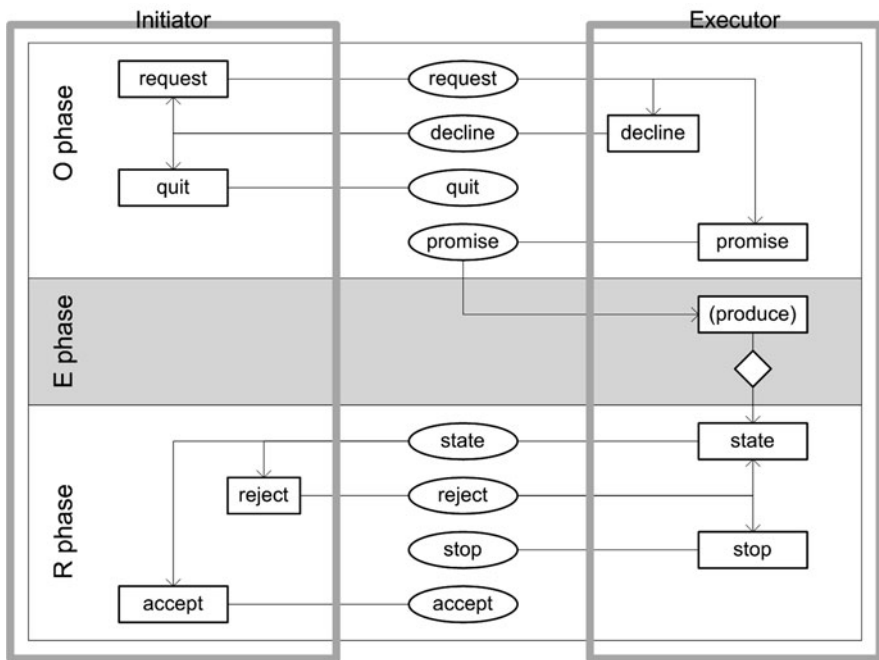


Fig. 18.8 Interaction pattern of a transaction according to (Dietz and Habing, 2004)

conversation that results in a commitment of the executor to perform the P-act (produce). In the second phase the P-act is actually done and a corresponding factum is established (“P-act has been executed”). The result phase is a factagenic conversation that aims at reaching an agreement on this factum to assess whether the commitment has been met.

The order phase starts with a request from the initiator to perform a certain P-act. Should the executor decline it, the initiator can decide to quit or to make another request, possibly after having convinced the executor on the discussion layer to consider his original request again, or after having realized the inappropriateness of his request and formulated a new one. If the executor accepts the request it becomes a promise, i.e. a binding commitment. The executor will then perform the P-act (execution phase) and enter the result phase by stating that he has done the P-act as agreed. The initiator can now reject this statement upon which the executor can stop (unsuccessful termination) or make another statement. If the initiator accepts the statement the whole transaction is terminated successfully.

18.3.2.3 The Composition Axiom

The composition axiom claims that each transaction is either activated externally (e.g. by a customer), or it is embedded in another transaction, or it is self-activated. In the first case the transaction is started by some person or organization in the environment, i.e. that lies outside the borders of the organization in question. Embedded transactions take into account that products and services typically have a hierarchical structure which must be mirrored by the transactions that produce them. Finally, a transaction can activate itself. This takes care of repeated activities, e.g. periodical transactions.

18.3.2.4 The Distinction Axiom

The distinction axiom states that all human activities, and thereby also business activities, can be divided into three categories: forma, informata and performata. Forma concerns the “outer appearance”. In the C-world this corresponds to passing on or receiving information. Informata is concerned with the content that is “in the form”. In the C-world it comprises the expression or interpretation of thoughts. Performata is related to the creation of new things with the help of the form, i.e. “through the form”. Regarding the C-world this could involve establishing and following up commitments (creation of commitments and facts). These levels apply also to the P-world.

18.4 Conclusion

We have introduced two different approaches to a domain ontology for business that mark the opposite ends of a plane spanned by scope and elaboration. Socio-Instrumental Pragmatism makes a weak ontological commitment that can be applied to any business context and even beyond that. But in order to do this

the concept system, the taxonomy and the relations have to be refined, and axioms or rules need to be introduced. Enterprise Ontology, on the other hand, makes a strong ontological commitment which might be in conflict with a particular view on a business. Its scope is therefore limited but in an appropriate context it can provide a strong support.

But in order for business ontologies to be really applicable they must support activities in a multitude of interrelated domains. No single ontology can achieve that. We therefore need mechanisms to connect related ontologies in an effective way. This implies that ontology X may import or derive concepts and relations from another ontology Y. But it also means that it must be possible to integrate ontologies from different but related domains to a new one that covers all these domains and resolves the conflicts that exist between them (see Section 18.1.2). It is the combination of the bottom-up and top-down approach that can bridge the gap between a general ontology for the business domain and the application-specific ontology.

References

- Anderson, P. 1999. Complexity theory and organization science. *Organization Science* 10(3): 216–232.
- Austin, J.L. 1975. *How to do things with words*. Cambridge, MA: Harvard University Press.
- Bertolazzi, P., C. Krusich, M. Missikoff. 2001. An approach to the definition of a core enterprise ontology: CEO. Paper presented at the International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations (OES-SEO 2001), Rome, 14–15 Sep 2001.
- Bugaite, D., and O. Vasilecas. 2005. Framework on application domain ontology transformation into set of business rules. Paper presented at the International Conference on Computer Systems and Technologies – CompSysTech 2005.
- Bunge, M. 1977. *Ontology I: The Furniture of the World*. Dordrecht: Reidel.
- Carmichael, D.J., J. Kay, and B. Kummerfeld. 2004. Personal ontologies for feature selection in intelligent environment visualisations. In *Artificial intelligence in mobile systems 2004*, eds. J. Baus, C. Kray, and R. Porzel, 44–51. Saarbrücken, Germany: Universität des Saarlandes.
- Corbett, D. 2003. Comparing and merging ontologies: A concept type hierarchy approach. In *Foundations of intelligent systems*, eds. N. Zhong, Z.W. Ras, S. Tsumoto, and E. Suzuki. Proceedings of the 14th International Symposium, ISMIS 2003. Maebashi City, Japan, 28–31 Oct 2003, 75–82. Berlin: Springer.
- Davern, M. 1997. Social networks and economic sociology. A proposed research agenda for a more complete social science. *American Journal of Economics and Sociology* 56(3):287–301.
- Dieng, R., and S. Hug. 1998. Comparison of <<personal ontologies>> represented through conceptual graphs. In ECAI 98. 13th European Conference on Artificial Intelligence, ed. H. Prade, 341–345. New York, NY: Wiley.
- Dietz, J.L.G. 2006. *Enterprise ontology: Theory and methodology*. Heidelberg: Springer.
- Dietz, J.L.G., and N. Habing. 2004. A meta ontology for organizations. In *On the move to meaningful internet systems 2004: OTM 2004 workshops*, eds. R. Meersman, Z. Tari, A. Corsaro, P. Herrero, M. S. Pérez, M. Radenkovic, V. Robles, C. Santoro, A. Albani, K. Turowski, M. Jarrar, A. Gangemi, E. Duval, P. Spyns, and A. Palinginis, Proceedings of the OTM Confederated International Workshops and Posters, GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004, Agia Napa, Cyprus, 25–29 Oct 2004, vol. 3292, 533–543. Berlin: Springer.
- Domingue, J. 1998. Tazebao and WebOnto: Discussing, browsing, and editing on the web. In eds. B. Gaines, and M. Musen. Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, 18–23 Apr, Canada: Banff.

- Embley, D., B. Kurtz, and S. Woodfield. 1992. *Object oriented system analysis: A model-driven approach*. Upper Saddle River, NJ: Yourdon Press.
- Farquhar, A., R. Fikes, and J. Rice. 1997. The Ontolingua server: Tools for collaborative ontology construction. *International Journal of Human Computer Studies* 46:707–728.
- Flood, R.L. 2005. Unleashing the “open system” metaphor. *Systemic Practice and Action Research* 1(3):313–318.
- Fox, M.S., and M. Gruninger. 1998. Enterprise modeling. *AI Magazine* 19(3):109–121.
- Gareth, M. 1997. *Images of organization*. London: Sage.
- Giddens, A. 1986. *The constitution of society. Outline of the theory of structuration*. Berkeley, CA: University of California Press.
- Goldkuhl, G. 2001. Communicative vs material actions: Instrumentality, sociality and comprehensibility. Paper presented at the 6th International Workshop on the Language Action Perspective (LAP2001), Montreal.
- Goldkuhl, G. 2002. Anchoring scientific abstractions – Ontological and linguistic determination following socio-instrumental pragmatism. Paper presented at the European Conference on Research Methods in Business and Management (ECRM 2002), Reading, 29–30 Apr 2002.
- Goldkuhl, G. 2005. Socio-instrumental pragmatism: A theoretical synthesis for pragmatic conceptualisation in information systems. Paper presented at the 3rd International Conference on Action in Language, Organisations and Information Systems (ALOIS), University of Limerick.
- Goldkuhl, G., and P. Ågerfalk. 2000. Actability: A way to understand information systems pragmatics. Paper presented at the 3rd International Workshop on Organisational Semiotics, Stafford, 4 July 2000.
- Goldkuhl, G., and P. Ågerfalk. 2005. IT artefacts as socio-pragmatic instruments: Reconciling the pragmatic, semiotic, and technical. *International Journal of Technology and Human Interaction* 1(3):29–43.
- Goldkuhl, G., and M. Lind. 2004. Developing e-interactions – A framework for business capabilities and exchanges. Paper presented at the 12th European Conference on Information Systems, Turku, Finland, 14–16 June 2004.
- Goldkuhl, G., A. Röstlinger, and E. Braf. 2001. Organisations as practice systems – integrating knowledge, signs, artefacts and action. Paper presented at the Organisational Semiotics, IFIP 8.1 Conference, Montreal.
- Gordijn, J. 2004. E-business value modelling using the e3-value ontology. In *Value creation form e-business models*, ed. W.L. Curry, 98–127, [Chapter 5](#). Oxford: Elsevier Butterworth-Heinemann.
- Guarino, N. 1998. Formal ontology and information systems. In *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS'98)*, 3–15. Amsterdam: IOS Press.
- Guizzardi, G., G. Wagner, N. Guarino, and M.V. Sinderen. 2004. An ontologically well-founded profile for UML conceptual models. In *Proceedings of the 16th International Conference on Advanced Information Systems Engineering, CAiSE 2004*, vol. 3084, 112–126, Berlin: Springer.
- Haase, P., A. Hotho, L. Schmidt-Thieme, and Y. Sure. 2005. Collaborative and usage-driven evolution of personal ontologies. In eds. A. Gómez-Pérez, and J. Euzenat. *Proceedings of the 2nd European Semantic Web Conference, Heraklion, Greece, 2005* Vol. 3532, 486–499, Heidelberg: Springer.
- Habermas, J. 1984. *The theory of communicative action 1 – reason and the rationalization of society*. Boston, MA: Beacon Press.
- Halpin, T.A. 2001. *Information modeling and relational databases*. San Francisco, CA: Morgan Kaufmann.
- Heflin, J., and J. Hendler. 2000. Dynamic ontologies on the web. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, 443–449, Menlo Park, CA: AAAI/MIT Press.
- Huhns, M.N., and L.M. Stephens. 1999. Personal ontologies. *IEEE Internet Computing* 3(5): 85–87.

- Jensen, M.C., and W.H. Meckling. 1976. Theory of the firm: managerial behavior, agency costs and ownership structure. *Journal of Financial Economics* 3:305–360.
- Kendall, J.E., and K.E. Kendall. 1993. Metaphors and methodologies: Living beyond the systems machine. *MIS Quarterly* 17(2):149–171.
- Klein, B., R. Crawford, and A. Alchian. 1978. Vertical integration, appropriable rents, and the competitive contracting process. *Journal of Law and Economics* 21:297–326.
- Law, J. 1992. Notes on the theory of the actor-network: Ordering, strategy and heterogeneity. *Systemic Practice and Action Research* 5(4):379–393.
- Luhmann, N. 1990. The autopoiesis of social systems. In *Essays on self-reference*, ed. N. Luhmann, 1–21. New York, NY: Columbia University Press.
- OMG. 2005. Unified modeling language: Superstructure. Version 2.0, August 2005, Needham, MA: OMG. Retrieved, June 20 2006, from <http://www.uml.org/>.
- OMG. 2006. Unified modeling language: Infrastructure. Version 2.0, March 2006, Needham, MA: OMG. Retrieved, June 20 2006, from <http://www.uml.org/>.
- Osterwalder, A. 2004. The business model ontology – a proposition in a design science approach. Ph.D. thesis, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC: 173.
- Rose, J., and M. Jones. 2004. The double dance of agency: A socio-theoretic account of how machines and humans interact. Paper presented at the ALOIS Workshop: Action in Language, Organisations and Information Systems, Linköping.
- Ross, S. 1973. The economic theory of agency: The principal's problem. *American Economic Review* 63(2):134–139.
- Scott, A. 1997. Modernity's machine metaphor. *The British Journal Of Sociology* 48(4):561–575.
- Searle, J.R. 1997. *Speech acts – An essay in the philosophy of language*. Cambridge, MA: Cambridge University Press.
- Searle, J.R. 1999. *Expression and meaning. Studies in the theory of speech acts*. Cambridge, MA: Cambridge University Press.
- Senge, P.M. 1990. *The fifth discipline. The art and practice of the learning organization*. New York, NY: Doubleday.
- Uschold, M., M. King, S. Moralee, and Y. Zorgios. 1998. The enterprise ontology. *Knowledge Engineering Review* 13(1):31–89.
- Verharen, E. 1997. *A language-action perspective on the design of cooperative information agents*. Tilburg: Katholieke Universiteit Brabant.
- Walsham, G. 1997. Actor-network theory: Current status and future prospects. In *Information systems and qualitative research*, eds. A.S. Lee, J.Liebenau, and J.I. Degross. London: Chapman & Hall.
- Williamson, O.E. 1981. The modern corporation: Origins, evolution, attributes. *Journal of Economic Literature* 19:1537–1568.
- Williamson, O.E. 1983. *Markets and hierarchies*. New York, NY: Free Press.
- Williamson, O.E. 1998. *The economic institutions of capitalism*. New York, NY: Free Press.
- Wittgenstein, L. 2001. *Tractatus logico-philosophicus*. London: Routledge.

Chapter 19

Ontologies for E-government

Knut Hinkelmann, Barbara Thönssen, and Daniela Wolff

19.1 Motivation

Serious efforts have been made for years to bring e-government forward. Although according to an e-government study 2006 nearly 50% of the examined services offered by EU states are fully available online (Capgemini, 2006) the majority of services on governmental web sites (especially of small municipalities) are in “stage 1” (information) or “stage 2” (interaction)¹. That is: providing online information about public services, e.g. opening hours or addresses of departments or provision of downloading of forms. Processing of forms, including authentication, (“stage 3” - two-way interaction) and mainly “stage 4” (transaction, that is full case handling, decision and delivery e.g. payment) is reached only for a few services. Taking into consideration that only 20 basic public services (12 for citizens, 8 for businesses) have been evaluated in the study (Chevallerau, 2005) – out of more than 1,000 (e.g. in Switzerland approximately 1,200 services have been identified (eCH0015, 2006)) – the necessity for new approaches is palpable.

There are recent developments and trends on various levels that can have significant influence on the progress of e-government:

- *architecture*: In a service-oriented architecture processes are supported by independent services that are made available on a network instead of having complex monolithic systems. They can be accessed via defined interfaces while implementation details are hidden to the user. This is particularly useful in administrative processes where various authorities are often involved.
- *technology*: web services are a technology for implementing service-oriented architectures in an internet environment. They provide standards for identification

K. Hinkelmann (✉)

University of Applied Sciences Northwestern Switzerland FHNW, Olten, Switzerland
e-mail: knut.hinkelmann@fhnw.ch

¹The classification is taken from IDABC (Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens) (Chevallerau, 2006)

and invocation of services. In particular, they offer the advantage of using common services instead of each administration having to implement each service itself.

- *semantics*: With the semantic web new standards for ontologies are developed for a machine-understandable representation of the semantics of (web) services, thus enabling the finding of relevant information as well as an intelligent discovery, composition, invocation, monitoring and maintenance of (web) services on the internet. Thus, ontologies can be seen as an enabler bringing service-oriented approaches to their full potential.

As public services are based on legal rules and regulations binding for all public authorities, various service providers (e.g. all municipalities) have a (broadly) similar service portfolio. However, neither naming nor service presentation at the particular portals, nor least of all service execution are similar. This challenges the citizen to find the same service provided by different municipalities and hinders the administrations' provision of one-stop-services or even integration. Ontologies are a promising answer to particular challenges in e-government:

- *finding services and information*: Ontologies contribute to a common understanding of service description. Information about services can be found on web pages of public authorities and on specific portals (e.g. www.ch.ch in Switzerland). Although most of the web pages use life events as a structuring principle, they differ a lot in naming and structuring of life events. A "life event" is understood as a special situation in a person's life, like marriage, childbirth, house building etc. that requires a public administration's services; similarly, "business situations" are defined for companies, reflecting their requirements such as applying for work permits or importing goods. Ontologies can contribute to a common understanding of a domain and allow for mapping different information structures, e.g. mapping of life events differently presented on different e-government portals or web sites.
- *process design and implementation*: As various service providers have broadly the same service portfolio, the sharing and reuse of experiences and domain knowledge can significantly reduce effort and increase quality of services. Initial steps are made with repositories of reference models, best practices and use cases. Adding semantic metadata to these resources improves the identification of relevant resources. In addition, enhanced modelling methodologies with explicit representation of design rationales and design decisions help a service provider to customize a process for a new context. Both semantic metadata and design decisions can be modelled as ontologies to provide the needed control.
- *interoperability and composition of services*: One stop e-government (Wimmer and Tambouris, 2002) makes it possible for any public authority to act as a front office for delivering e-government services regardless of the administrations actually involved. Semantic Web Services enable interoperability of administration by automatic discovery and composition of appropriate web services. Consider, for example, the service "change of residence". In one-stop e-government the citizen can start the process (either at a portal or on the web site of his/her old

or new municipality) and the registration and deregistration services of the concerned municipalities have to be identified automatically. Exploring the semantic metadata is the initial step of that approach.

- *flexibility and adaptivity of process execution*: Web services are basic building blocks for building online processes. Semantic process models combined with inference rules and integrity constraints facilitate self-adaptivity and flexibility of process execution, allow for context-dependent resource allocation and ensure that constraints and guidelines are satisfied.
- *Maintenance and change of services*: As public services are based on legal rules and regulations, any change of these regulations can cause adaptation of processes. To ensure compliance of processes with regulations, the affected process and activities can be found easily if a law or regulation changes. This can be achieved by explicitly representing and inferring the reasons for design decisions.

In recent years ontologies have become common for semantically enriched formalization of knowledge. [Chapter 2](#) provides a framework for modelling knowledge with ontologies in the e-government domain completed by examples of efforts already undertaken in our research projects covering all of the abovementioned challenges. Even though the e-government domain is a very important application area, our approach is not limited to it but can be used in any business domain. However, the proceeding introduced has been used in several e-government projects and proofed eligible.

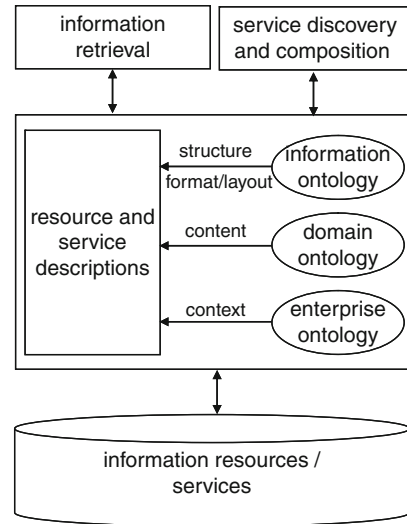
19.2 State of the Art in E-Government Ontologies

Several European countries are currently working on a common service description aiming for a consistent presentation on portals run by the various service providers and establishing a basis for realising one stop e-government. With this approach e-government is no longer regarded as a portal where public administrations publish “passive” information but as an active source for knowledge and service sharing. In addition it is perceived that knowledge sharing is no longer limited to humans but include software agents. As ontologies are supposed to be representations of the agents’ agreements about the set of concepts that underlie the information to be shared (Gruber, 1993b) they are an appropriate instrument of e-government knowledge representation (cp. (Gugliotta et al., 2006)).

In (Abecker et al. 1998) a methodology for organizational memories has been presented that distinguishes three ontologies (see Fig. 19.1):

- The *information ontology* describes the different kinds of information resources with their respective structure and format properties. The vocabulary for the information resource metamodels comes from the information ontology.
- The *enterprise ontology* defines the context in which information resources are used and generated. The top-level of the enterprise ontology defines a meta model for processes or organisational structure

Fig. 19.1 Types of ontologies (Abecker et al. 1998)



- The *domain ontology* defines concepts modeling the content of information resources and services. It is obvious that in particular the domain ontology is specific for each new application area.

Throughout our lives, we all experience many events such as birth, marriage, building a house, retirement, and even losing one's wallet. The "life" of a business also follows a series of predictable events such as the registration of a new business, relocation of a business or government forms and reports associated with various stages in a business' life. These events are called "life events". To help citizens and businesses deal with these kinds of events, most of the administration portals or web sites that have gathered information resources and services have a navigation structure that corresponds to these life events. In this sense, life events are organized as taxonomies – a simple form of ontologies (Daconta et al., 2003). Analysis of various portals and web sites has shown, however, that there is no consensus on what these life events should be called and structured.

In the SmartGov project a combined enterprise and domain ontology was built that provides a conceptual description of e-government services (Fraser et al., 2003). A representation in RDF is a cyclic graph and can become quite complex. This can cause problems both for public authority staff to maintain the ontology and also for user navigation. To avoid these problems in SmartGov the structure of the ontology was simplified in the form of a directed acyclic graph. This graph was extracted directly from the ontology, starting from a set of relevant top-level concepts that adequately describe public authority service provision. The top-level concepts are *activities, actors, issues, legislation, needs, process, requirements, responsibilities, results, rights* and *service types*. The subsequent structure has been engineered to enable searchability and function. Each concept has an optimal number of children,

which represent the domain as accurately as possible, while at the same time creating a logical search path to give an unambiguous route to the desired target (Fraser et al., 2003).

The Terregov project makes use of semantic technologies for achieving interoperability and integration between e-government systems. Rather than providing ontologies they implemented ontology creation and storage tools to allow ontologies to be created by domain experts (Wroe, 2005).

In the OntoGov² project we defined the top-level structure of e-government domain ontologies including life events and legal concepts as well as service and life-cycle ontologies (Stojanovic et al., 2006). The life event ontology was based on the life event structure of the national web site of Switzerland³ (www.ch.ch). The idea, however, was not to define a standard ontology for life events but to provide a general structure that can be reused and adapted to specific applications and that enables information exchange and interoperation between web services of different service providers (for problems associated with reusing the life event ontology see Section 19.3.3).

As can be seen, there is no standard domain ontology – even for life events a consensus will barely be enforceable. Potentially, for every application a new ontology has to be built and maintained.⁴

19.3 Ontologies to Formalize a Shared Understanding of Meaning

Even though there is an agreed definition of an ontology as a “formal specification of a shared conceptualisation”, as stated by Gruber (1993a) and refined by Borst (1997), the meaning of “conceptualisation” can be regarded differently (cp. (Pinto and Martins, 2004)). Here we follow the view of Pinto and Martins where conceptualisation is regarded as “an abstract conceptual model for the knowledge to be represented in the ontology” (Pinto and Martins, 2004). To bridge the gap between knowledge, phrased ambiguously in natural language (e.g. in regulations) and its unambiguous representation in an ontology we introduce the intermediate stage of semi-formalization.

Since the early 1990s there has been a lot of research on ontology design and creation (amongst others Gruber (1993b), CommonKADS (Schreiber et al., 1999) or Ontology Development 101 (Noy and McGuinness, 2002)). However, no standard methodology for ontology building exists.

²OntoGov (Ontology Enabled E-Gov Service Configuration) is a project funded in the IST Programme of the European Union (IST-2004-27090). For further information consult <http://www.ontogov.com/>

³The Swiss Federal Chancellery was a partner in the OntoGov consortium

⁴Application here is understood as any kind of software developed to execute public services (using semantic technologies)

As ontology development is part of an IT project, it has to be embedded into a project framework. We suggest using the waterfall model as it is widely accepted for IT projects in the public sector (e.g. HERMES, which follows the waterfall model, is the stipulated IT project management method in Switzerland (FSUIT, 2004)). Figure 19.2 depicts our ontology development process.

In the first phase, the pilot study, the purpose and goal of the project should be investigated with respect to the business strategy. The various possibilities of knowledge formalization should be considered to ascertain that ontologies are the appropriate model. This is variously called “specification” (Pinto and Martins, 2004) or “capture motivating scenarios” in TOVE (Gruninger and Fox, 1995) or “identify purpose and scope” in ENTERPRISE (Uschold, 1996, Uschold et al., 1998) or “requirement specification” in METHONTOLOGY (Fernández et al., 1997).

Within the concept phase ontology development is performed comprising three levels of formalisation: informal (knowledge is captured in natural language), semi-formal (knowledge is represented in a semi-formal way e.g. in structured templates) and formal (knowledge is strictly formalized in OWL,⁵ SWRL⁶ and OWL-S⁷). Therefore ontology conceptualisation, formalization and implementation are conducted in this phase.

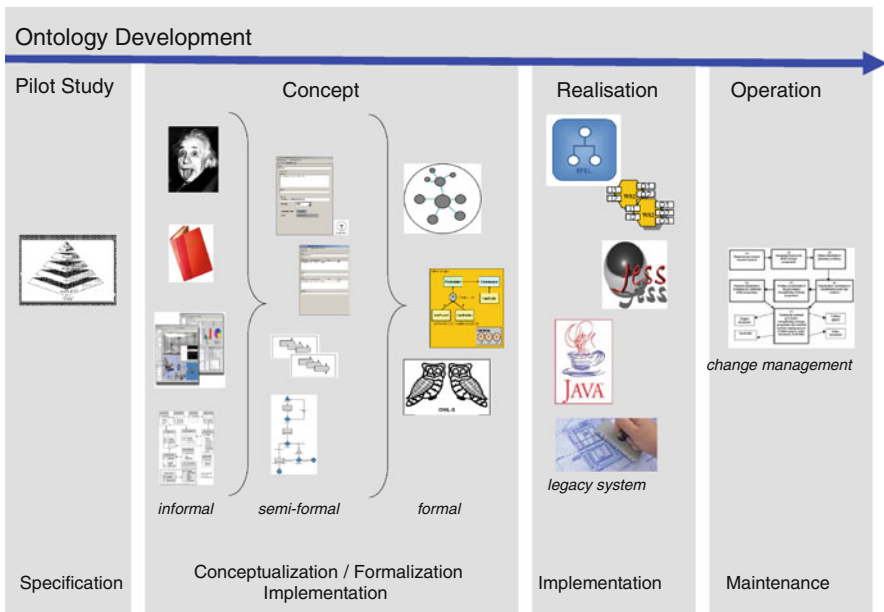


Fig. 19.2 Ontology lifecycle

⁵OWL: Web Ontology Language (McGuinness and vanHarmelen, 2004)

⁶SWRL: Semantic Web Rule Language (Horrocks et al., 2004)

⁷OWL-S: Web Ontology Language for Services (Martin, 2004)

The realisation phase is about implementation of ontologies, that is making ontologies accessible out of program code (e.g. legacy systems or web services). This can be achieved by migrating it into Java code, thus implementing application programming interfaces (APIs) to access the ontology via ontology management systems like Protégé⁸

Maintenance is performed in phased operations comprising ontology management like updates (add or delete concepts or properties), merging or integration of ontologies (q.v. Section 19.3.3 of this chapter).

In the following we focus on the concept phase with its three levels of formalization (informal, semi-formal and formal). We need the formal level to make ontologies machine understandable. In most cases, however, a domain expert will not be able to transform informally described knowledge (e.g. guidelines written in natural language) into a formal representation like OWL, using Protégé as ontology management tool. To bridge the gap we introduce the semi-formal level where domain knowledge, services and rules are formalized in a way domain experts are able to phrase and IT staff is able to transform without missing or misinterpreting business logic.

19.3.1 Starting with Terms

In general there are two ways to build ontologies: either from scratch or by reusing existing ones. Gugliotta (2006) gives a detailed overview of current (research) activities in this field. Please refer amongst others to Peristeras and Tarabanis (2006) for “top-level reusable models for the overall e-government domain”.

In the document at hand we talk about ontology development from scratch but starting not immediately with concept modelling but with *term*⁹ definition. What Oscar Chappel stated for rule modelling can be adapted for ontology modelling: “everything starts with terms” (Chappel, 2006), that are candidates for main concepts. Sources for terms can be “anything” that reflects business behaviour:

- documents
 - lists of services and their descriptions (eCH, 2006)
 - theme-catalogues (e.g. lifecycle aspects, business situations (eCH, 2008)
 - handbooks
 - guidelines
 - policy papers
 - glossaries

⁸Protégé is a free, open source ontology editor and knowledge-base framework (<http://protege.stanford.edu>).

⁹Merriam Webster Online: Definition 4a: a *term* is a word or expression that has a precise meaning in some uses or is peculiar to a science, art, profession, or subject <legal terms>, URL: <http://www.m-w.com/cgi-bin/dictionary>

- data
 - meta-data, descriptors, identifiers
 - key-words for search engines
 - database entity and attribute names
- mind
 - expressions passed on orally.

When questing for terms, the goal and scope of the ontology to be built should be kept in mind. Not every term written in a handbook should become a concept candidate even though every main concept supposed to be in the ontology should be explicitly represented.¹⁰ Therefore *relevance* is an eligibility criterion for ontology inclusion as the list of terms could quickly become complex.

Within the public sector – especially in federal states – finding and defining relevant terms is even more complex as there is no “authorized decision maker”. Several terms will be used for the same circumstances and one term will be used to describe different facts. As we will see later, ontologies provide a good solution for handling synonyms and homonyms without offending sensibilities. In the term building phase the various terms will be collected, their relation or specific meaning identified and documented.

There are several approaches for automated ontology creation (amongst others TextToOnto,¹¹ supporting semi-automatic creation of ontologies by applying text mining algorithms (Maedche and Staab, 2004)). This could be a good starting point for *term capturing* as ontology building is time consuming. As ontologies are complexity prone, ontology creation can be machine supported. However, a lot of manual labour is still necessary.¹²

Regardless of which approach is chosen the following attributes¹³ should be defined for every term (they will become data properties in the ontology):

- Manually defined attributes
 - label (the human-readable label including language information)
 - definition (a statement that represents the concept and essential nature of the term)
 - source (where the term has been taken from, e.g. out of a handbook)¹⁴

¹⁰*completeness* is a criteria very difficult to prove (cp. (Gómez-Pérez, 2001))

¹¹The system is freely available and can be downloaded at <http://sourceforge.net/projects/texttoonto/>

¹²An interesting topic for research is how term and fact modeling could be automated (as it is less formal) and how the semi-formal representation could then be used as input for automated ontology creation

¹³The attributes “label”, “definition” and “date issued” are attributes, “source” and “creator” are defined as meta-data terms by the Dublin Core Metadata Initiative (2006)

¹⁴In case the terms are extracted automatically this attributes can be created automatically, too

- Automatically created attributes
 - date issued
 - creator (e.g. domain expert who records the term).

The result of this step is a collection of terms with their attributes. Again, there are several methods of presenting the terms (as well as concepts), either as mind-maps (as introduced by Sure et al. (2002)) or simply as a (flat) list – where appropriate in alphabetical order.

We propose a repository where concepts can be stored in a structured but semi-formal way understandable by IT staff *and* by domain experts. An example of a structured term representation is given in Fig. 19.3¹⁵ for the term “Application”. A term is represented graphically as a named circle (labelled with a “T” for term);

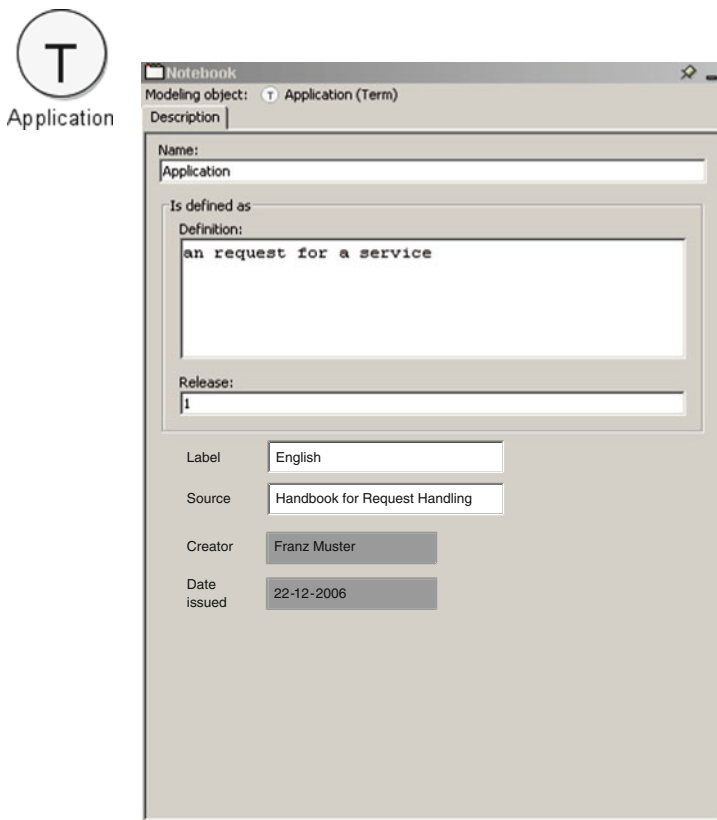


Fig. 19.3 A concept and its properties

¹⁵The interface is adapted from the FIT Buildtime for Adaptive Processes developed by BOC Asset Management (<http://www.boc-eu.com>) within the FIT project.

its attributes are captured in a so-called “notebook”. Several user workshops (performed within the FIT project¹⁶) have shown that this simple formalization is much easier to understand than using an ontology modelling tool like Protégé from the very beginning. As Macias and Castells (2004) pointed out: “Generally speaking, emerging web-based technologies are mostly intended for professional developers. They pay poor attention to users who have no programming abilities but need to customize software applications”.

After capturing the terms (along with their attributes) they can be grouped, initially by considering relationships between terms. Figure 19.4 shows a grouping of terms based on their graphical representation. Business people group terms along business aspects: life cycle events (leading to the group of “Environment&Construction”), project management issues (leading to the group of “ProjectAspects”), elements of business tasks (leading to the group of “Application Handling”) or stakeholder (leading to the group of “People”). Even though business people are not familiar with relating concepts, the two different kinds of groups in the example are easy to understand:

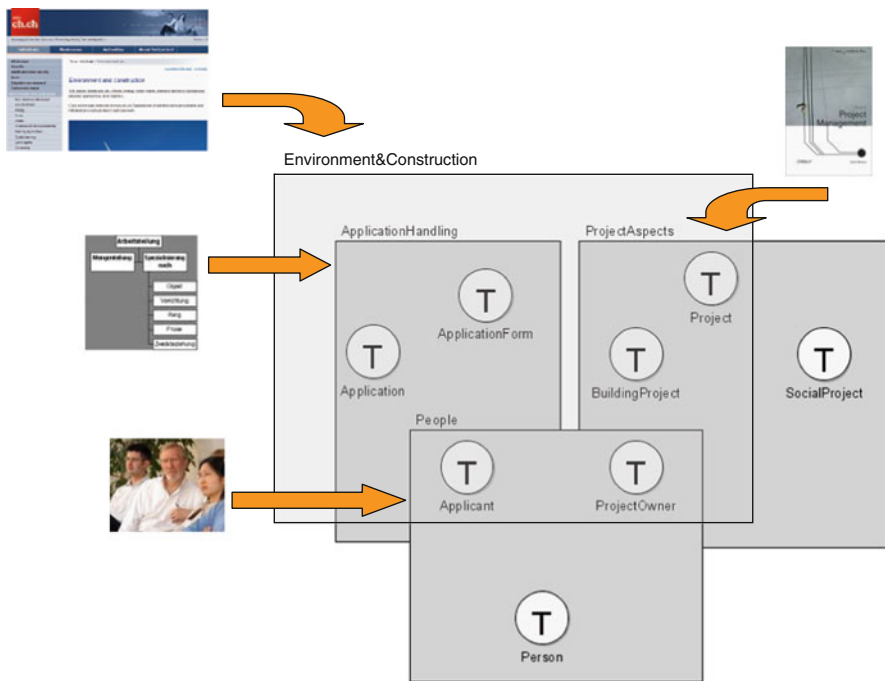


Fig. 19.4 Visual grouping of concepts

¹⁶FIT (Fostering self-adaptive e-government service improvement using semantic technologies) is a project funded in the IST programme by the European Union (IST-2004-27090). For further information consult <http://www.fit-project.org>

- (a) terms and specialisations of terms as in the *ProjectAspects group* (where the terms “BuildingProject” and “SocialProject” are specialisations of the term “Project”) and
- (b) terms related because of a specific business topic as the life event “Environment&Construction” (where the terms “Application”, “Application Form”, “Applicant”, “ProjectOwner”, “Project” and “BuildingProject” belong together).

Having grouped the terms, the next step is to make the relations explicit. Again we recommend a semi-formal way of building *facts* (Fig.19.5) that is, relating terms either in an hierarchical way (a “BuildingProject” *ISA* “Project”) or as a network (a “Application” *is based on* an “ApplicationForm”, a “ApplicationForm” *is the same as* a “RequestForm” etc.) The result of this step is a semi-formal *term model*.

Modelling facts is the last step in ontology building that can be done in such a convenient way for domain experts. Adding more properties, e.g. to express constraints for domain or range restrictions, would quickly make the forms complex and would lead to a kind of “ontology modelling simulation”.

It turned out that collecting terms and modelling facts is an iterative process. A good possibility is to start with terms extracted from standards (for example the

The screenshot shows a software window titled "Notebook" with a tab for "Applicant2Person (Fact)". The interface includes a "Description" section with the following fields:

- Name:** Applicant2Person
- Term 1:** A table with columns: Model directory, Model, Model type, Object, Class. The row contains: /Models/04 FI..., Rule Lev..., Applicant, Term.
- Relation:** IS A
- Term 2:** A table with columns: Model directory, Model, Model type, Object, Class. The row contains: /Models/04 FI..., Rule Lev..., Person, Term.
- Release:** 1

Below the form is a summary table:

Term	Relation (IS A / VERB)	Term
a Citizen	lives	at a Place of Residence
an Applicant	IS A	Person
an Application	is based on	an Application Form

Fig. 19.5 Relating terms to build facts

documents published by eCH for Switzerland). Having consensus on that specific terms invented by communities can be considered and incorporated.

19.3.2 Transforming Terms and Facts to Concepts and Properties

To transform terms and facts to concepts and properties we suggest that IT staff or knowledge engineers, who are familiar with ontology modelling, take the semi-formal models of terms and facts as input for formal ontology modelling, e.g. using a specific ontology modelling tool like Protégé. While the transition from terms to concepts (attributes to data-properties) and from facts to relations is currently performed manually, automation is planned.

During ontology modelling concepts may be added (e.g. to build a super-concept), data-properties will be completed (e.g. for cardinality) and properties will be extended or newly set up (e.g. relations between terms reflecting business aspects, domain and range constraints or simply stating synonymy).

Therefore the ontology model will probably differ from the term model. To ensure consistency between the ontology model and the term model “reverse engineering” is recommended. Transforming the ontology model “back” to the term model allows business people to continue with modelling business from their point of view. Using OWL as an interchange format,¹⁷ BOC’s modelling tool, for example, allows the import of ontologies and its representation as a term model.

As in the majority of cases public administration officers will not model the ontology on their own, a graphical presentation will make the cooperative development by business analysts or IT staff and domain experts less difficult (Fig. 19.6).¹⁸

19.3.3 Negotiating Reuse

It cannot be expected that one e-government ontology can be built for the entire public administration of a state let alone across borders. More likely, several ontologies will be available and could be joined, if reasonable, in two ways: fusion/merging or composition/integration. A lot of research has been done in the field of ontology reuse (cp. amongst others Pinto and Martins, 2002; Kalfoglou and Schorlemmer, 2005) and several tools have been developed to support the process.¹⁹

It is beyond the scope of this document to go deeper into that topic. However, as a rule of thumb it can be said, that merging ontologies is preferred if there

¹⁷An interchange format is a format that allows transformationen from one model to another without loss based on agreed standards.

¹⁸For ontology modeling Protégé is taken; for graphical presentation the Ontoviz tab is activated

¹⁹A comprehensive overview is given by the IOWA State University, Department of Computer Science: http://www.cs.iastate.edu/~baojie/acad/reference/2003-07-09_dataint.htm (Date 20-12-06)

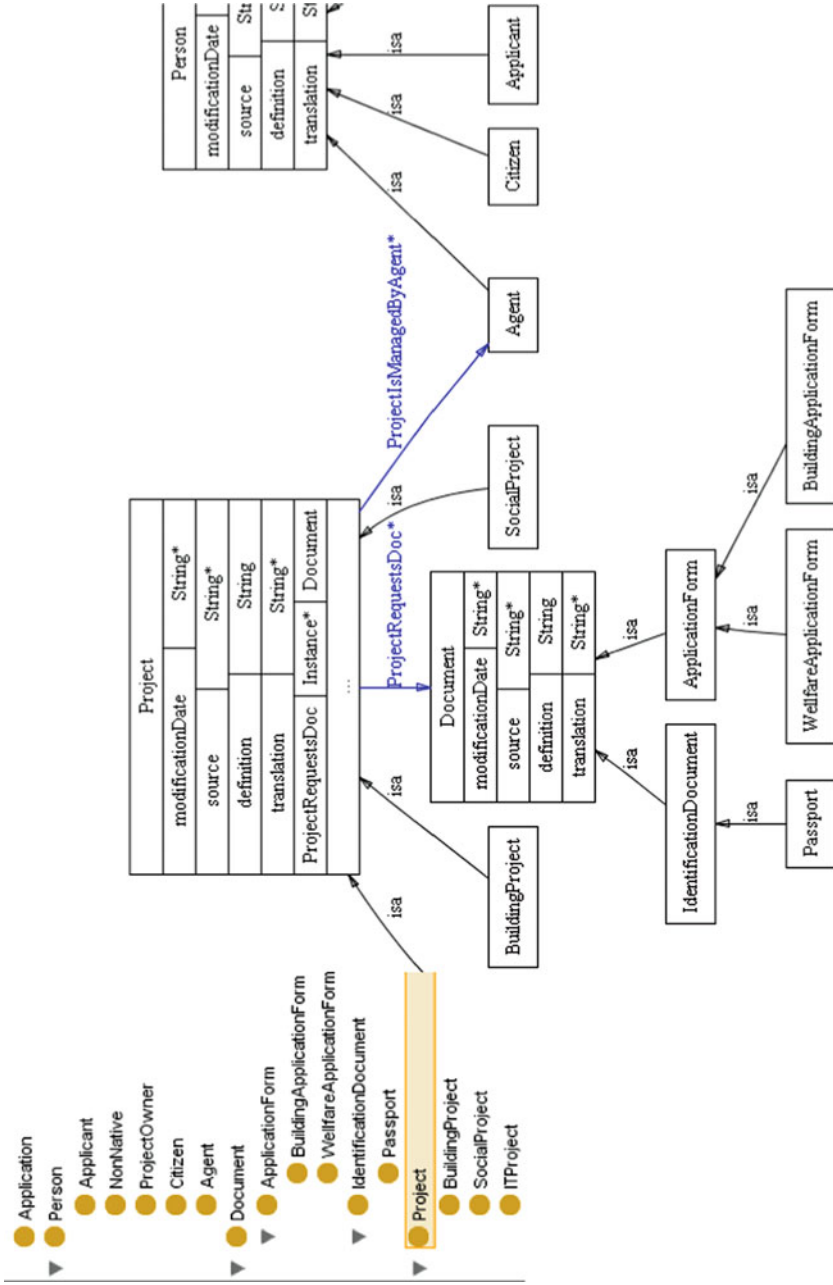


Fig. 19.6 Graphical presentation of concepts and properties

is a significant overlapping of concepts whereas integration is favoured if the overlapping is restricted.

Regarding the e-government domain, reuse of ontologies seems to be a promising approach. For example the “life event concept” is implemented on most governmental web sites worldwide. As already mentioned in Section 19.2, in the OntoGov project an ontology has been built based on the “life event concept” of the national web site of Switzerland (<http://www.ch.ch>). It was assumed that the ontology could be reused in any municipality in Switzerland or even in other countries. When verifying this intention, however, significant differences have been detected with respect to

- point of view (what has been defined as a life event concept in the ontology has not been regarded as one in an municipality and vice versa)
- structure (what has been modelled as super-concept in the ontology has been regarded as sub-topic in an municipality and vice versa)
- granularity (in some municipalities the life event structure is very detailed whereas in others it is not).

Reusing/joining ontologies is in any case not only a technical demand but requires expertise (by domain not IT experts) to negotiate the goal, scope and content of the new ontology. Public administrations move in the direction of knowledge sharing (and reuse) but federalism is still pronounced hindering the possibilities the technique already provides.

From this point of view integration seems to be the better approach for reuse. However the mentioned problems cannot be overcome simply and further research is needed for an applicable solution.

19.4 Ontologies for Modelling Semantically Enriched Processes

In Section 19.3 of this chapter we introduced a method for getting to (main) concepts and relations of a domain ontology. In the following section we will focus on metadata related to the business processes.

Although municipalities provide virtually identical services, implementation of these services takes place individually and is continually repeated.²⁰ For example: nearly every municipality in Switzerland (about 2,700) performs the service of registration (for someone moving in) and deregistration (for someone moving away). Even though there are efforts to standardize the processes, the real implementation differs significantly, due to a range of factors including the different IT infrastructure of the municipalities. The same is true for services of federal states and administrations like courts.

²⁰For example: the public service “Anmelden/Abmelden” is performed by ne

The exchange of experience and sample process models between similar institutions could drastically reduce the effort of service implementation. For example, in Germany the “Virtual Community Geschäftsprozess-Management²¹” (virtual community for business process management) is based on a repository of process models. Members can provide experience in the form of process models and profit from the experience of other members.

A similar approach is the propagation of reference models for e-government services and organisational structures that can be used and customized directly by municipalities and federal states. Reference models are abstract models of a domain of interest that represent best practices. As administrative organisations – as opposed to companies – do not have to protect a competitive advantage and their processes are based on legal foundations, e-government services are well suited for reference models.

Nowadays, repositories of (reference) models mainly use textual descriptions; thus, they are hard to quest and only indirectly reusable. Adding semantic metadata to process models would be much more helpful in finding adequate services and reusing the *models* (instead of their description) – an approach that is widely accepted for semantic web services. This, however, is still an open research issue, in particular as no standards for e-government ontologies exist.

Even without reference models, business process management in general must cope with the problem of life-cycle management, i.e. how to ensure compliance of models with regulations and how to customize process models. Adding not only metadata to a process but also information about design decisions is an approach for dealing with the problem.

Virtually all public administration processes are based on legal foundations or decrees that are derived from them. Therefore, such regulations also determine the design of a process. If a law is modified, all processes that are based on this law have to be checked for compliance. Besides legal aspects, organizational or technical reasons can also determine the design of a process. If one administration wants to reuse and customize a process from another administration it can be important to know what is specific to the administration and thus can be modified and what is vital for the process to work.

In Hinkelmann et al. (2006a) we presented an approach for life-cycle management of e-government services. To ensure consistency of services, all the decisions are documented. For each design decision at least one reason must be given. A decision includes a description in natural language and a formal reference to the elements forming the basis of the decision (e.g. the respective law). Analogous to IBIS (Issue Based Information System, (Kunz and Rittel, 1970), a design decision is regarded as a topic which is based on a reason. As a design decision can possibly lead to other design decisions, a line of argumentation results. Figure 19.7 shows the structure of a line of argumentation according to the graphic notation gIBIS (Conklin & Begeman, 1998, pp. 140–152).

²¹ <http://www.fhvr-berlin.de/vc-gpm/> (information in German only).

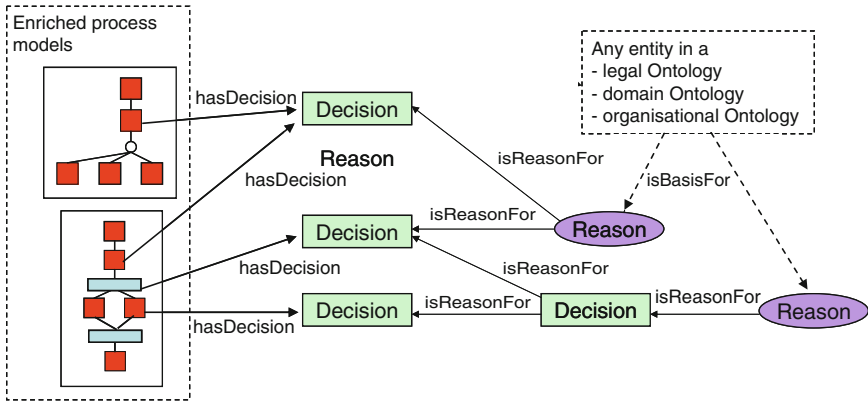


Fig. 19.7 Extending process models with life-cycle information

Apart from the natural language documentation, it is vital for knowledge-based support to also formally describe the design decisions and reasons. This is done by linking design decisions and reasons using an ontology in which the reasons for the design decisions are explicitly modelled and refer to the underlying law. This ontology is called “lifecycle ontology” as all activities performed on process models are

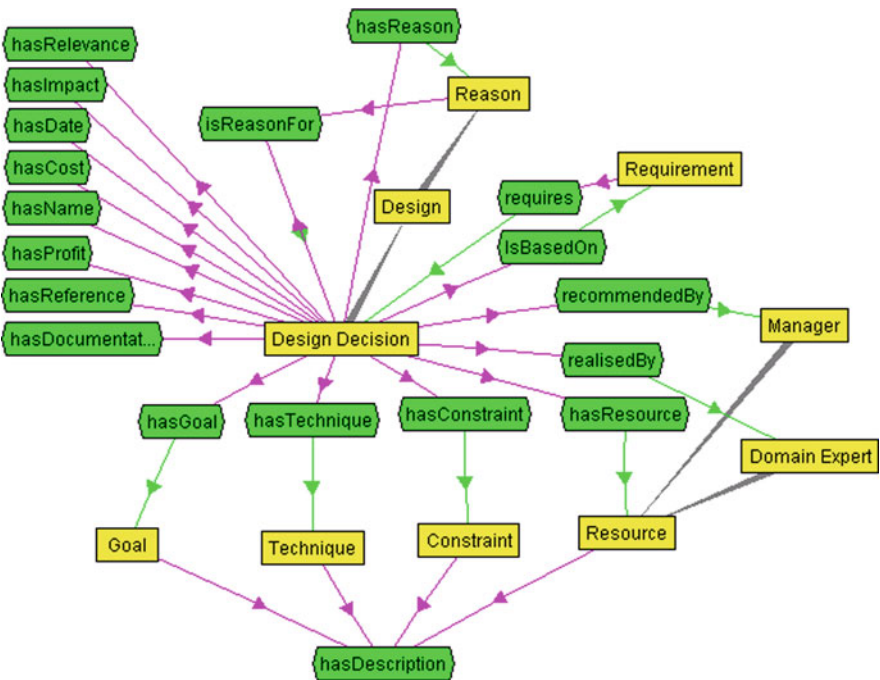


Fig. 19.8 Part of the OntoGov lifecycle ontology (developed using KAON)

captured, e.g. when adding a new activity to a process the reason must be expressed and the respective decision is saved along with further informations like user or creation date. Figure 19.8 shows the relevant part of the lifecycle ontology. The rectangles represent concepts and the hexagons represent relations. It can be seen that DesignDecision is a subconcept of Decision and that Decision is a subconcept of Reason, because every decision can have consequences, and so can be a reason for another decision. In this way we can model the decision-reason chains shown in Fig. 19.7.

In addition, extended process models are formally linked with the design decisions of the lifecycle ontology (see Fig. 19.7). In this way, design decisions and reasons that define process (or reference) models become transparent and traceable.

Details on how the design decision can be used for change propagation can be found in Hinkelmann et al. (2006a).

19.5 Ontologies for Modelling Business Rules

As already shown, public services are based on legal rules and regulations binding for all municipalities but at the same time municipalities may maintain distinctive features, so the business processes are quite *similar* but not *identical*.²² Additionally, a lag time can occur between adoption and implementation of rules, so delays and human errors can arise.

Second, e-government services are knowledge-intensive processes resulting in manually performed process execution. Even though ICT supports most of the *functions* (e.g. tax system) very few *tasks* are automated. In addition, it is not only the knowledge about how to perform *the tasks* within a process (the so called functional knowledge) but also the knowledge about *the process flow* (the so called process knowledge) that isn't explicitly documented.

So the next step in enhancing business processes is adding business rules. Business rules can be regarded as an appropriate approach not only to make that knowledge explicit, to have greater control and oversight, but also to ensure consistent implementation of policies in an automated way, because changes to the rules can be immediately reflected in all services related to them.

19.5.1 Business Rules Classification

The Business Rules Group defines a "Business Rule" as:

...a statement that defines or constrains some aspect of the business. It is intended to assert business structure, or to control or influence the behavior of the business" (Business Rules Group, 2006).

²²The terms *service* and *process* differ in their coverage: whereas *service* comprises all aspects of e-government service provision a Public Administration has to offer, *process* is about the (IT-supported) tasks performed within a *service*.

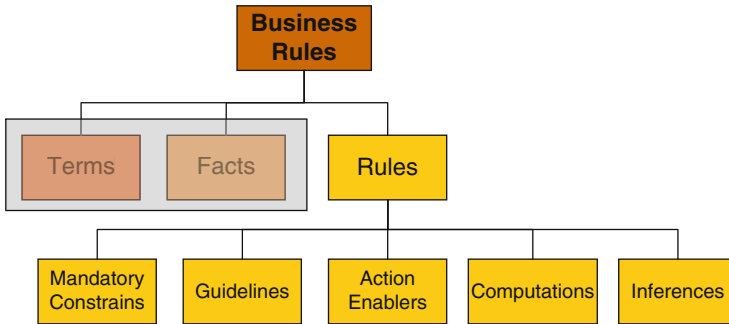


Fig. 19.9 Classification scheme after Barbara von Halle (2002)

There are several classification schemata for Business Rules formalization. Since it is well understood and comprehensive, we follow the classification scheme given by von Halle (2002, p. 27 ff.) in order to explain the different kinds of business rules.

Business rules consist of terms, facts and rules. “The terms and facts are the semantics behind the rules. They will also become the foundation for a logical data model and physical database [...]” (von Halle 2002, p. 32). In Section 19.3 we already explained how terms and facts can be identified and transformed to concepts and properties (Fig. 19.9). Rules are split up into five sub-classes:

- mandatory constraints
- guidelines
- action enablers
- computation rules
- inference rules

In the following discussion we explain these rule types in more detail. Similar to the approach for determining terms and facts we follow a two-step approach starting with a semi-formal representation that can easily be understood by people in public administrations and that can later be formalized in cooperation with IT people and knowledge engineers.

19.5.2 Semi-Formal Rule Representation

Semi-formal representation is the starting point for rule development. Each rule type has a specific structure or uses predefined relations. The rules should be phrased using previously defined terms and facts (see Section 19.3). It can also be the case, however, that new terms and facts are identified while formulating the rules thus leading to an extension of the ontology.

Mandatory Constraints

Mandatory Constraints are statements which must always be applied. To express this kind of rule the auxiliary verb “must” is used as a predicate. The phrase “must not” is used to express negative constraints.

Subject	Predicate	Object
a resident	MUST provide	a lease contract or contract of purchase for her place of residence
a decision	MUST BE in list	application decision ('applicaiton approved', 'application denied', 'decision postponed')

Guidelines

Guidelines are rules which could be followed but are not mandatory. Their form is similar to mandatory constraints but using the auxiliary verb “should” or “should not” to express the negation

Subject	Predicate	Object
a confirmation	SHOULD BE sent	within 5 days
a decision	SHOULD include	an explanation

Action Enablers

Action Enablers initiate a process (step), if the condition holds.

	Term	Condition		Action
IF	a building	is closer than 50 meters to a natural water	DO	approve environmental compatibility
	a building	is under protection of historical heritage		historical preservation agency approves application

Computations

Computations are rules to perform calculations, like sum and difference.

Subject	Computation
fee	IS COMPUTED AS days of delay x 10 CHF

Inferences

Inferences are statements which establish the truth of a new fact, if the condition holds.

	Term	Condition		Consequence
IF	an application a building	is delayed is older than 100 years	THEN	a fee is payable it is an historical building

19.5.3 Formalization

To automatically apply rules, a formal, executable rule language is required for all kinds of business rules. There is a broad spectrum of formalisms with different levels of expressiveness. Terms and facts, for instance, can be represented as vocabulary, database schema or ontologies. In [Chapter 3](#) we introduced modelling terms and facts as concepts and properties. To formalize rules it makes sense to use the ontology as knowledge base. In this section we describe how to express business rules using ontologies.

19.5.3.1 Property Restriction

Some mandatory constraints can be expressed by constraining the range of a property (value restrictions) or the number of values a property can take (cardinality restrictions).

Subject	Predicate	Object
a decision	MUST BE in list	application decision ('applicaiton approved', 'application denied', 'decision postponed')

This sample rule can be expressed by using a value restriction. The concept “decision” provides the property “hasApplicationDecision” which contains a list of possible values (Fig. 19.10).

The following document fragment expresses the same restriction in OWL:

```

<owl:Class rdf:ID="decision">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">

```

```

        <applicationDecision rdf:ID="Application_denied"/>
        <applicationDecision rdf:ID="Application_approved"/>
        <applicationDecision rdf:ID="Decision_postponed"/>
    </owl:oneOf>
</owl:Class>
</owl:allValuesFrom>
<owl:onProperty>
    <owl:FunctionalProperty rdf:ID="hasApplicationDecision"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<!--abbreviated -->
</owl:Class>
    
```

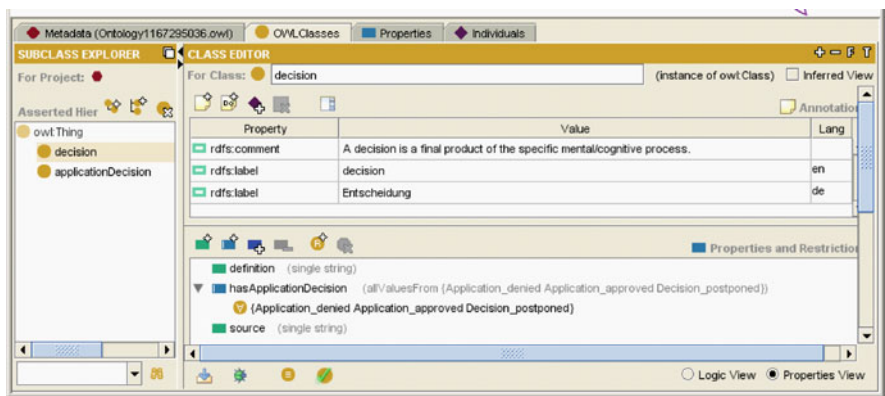


Fig. 19.10 Value restriction modelled in Protégè

19.5.3.2 Semantic Web Rule Language

Restrictions cannot be used to express every kind of constraint; in particular they cannot be used to express inferences, computations and guidelines. Therefore we use the Semantic Web Rule Language (SWRL) which has been published as a Member Submission by W3C. The aim of SWRL is to combine the Ontology Web Language OWL DL and OWL Lite with the Datalog RuleML sublanguage (Horrocks et al., 2004). It thus extends the set of OWL axioms to include Horn-like rules and enables Horn-like rules to be combined with an OWL knowledge base.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). Whenever the conditions specified in the antecedent hold, the consequent must be true. Antecedent and consequent consist of zero or more atoms. Atoms in these rules can be of the form $C(x)$, $P(x,y)$, $sameAs(x,y)$ or

differentFrom(x,y), where C is an OWL concept, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values (Horrocks et al., 2004).

Inferences

Inferences can be expressed using SWRL. If a condition holds, then the fact in the consequence must hold, too. In ontologies e.g. a value of a property is derived.

	<u>Term</u>	<u>Condition</u>		<u>Consequence</u>
IF	a building	is older than 100 years	THEN	it is an historical building

To express the inference written above the term “a building” can be presented as a concept, which contains the datatype property “age”. To present the consequence, a concept called “ancientBuilding” has to be added, which can be modeled as a sub-concept of “building”.

```

building(?x) ^
age(?x, ?y) ^
swrlb:greaterThan(?y, 100)
  → historicalBuilding(?x)
    
```

If the condition “an instance of building exists, whose age is greater than 100” holds, the found instance is also an instance of “ancientBuilding”.

Mandatory Constraints

Mandatory constraints which cannot be represented as property-restrictions can be expressed by using SWRL. Consider for example the mandatory constraint below.

<u>Subject</u>	<u>Predicate</u>	<u>Object</u>
a person	MUST be	Older than 18 to make an application

Several concepts have to be created to be able to present the rule. First the two terms “person” and “application” have to be modeled as concepts. To present the fact “an application is filled out by a person” the object property “filledOutByPerson” is defined with the domain “application” and the range “person”.

The constraint can be expressed by a rule using negation stating that there is an inconsistency if the age of an application is not 18 or older. Thus, we use

the concepts “exception”, “ontology” and “ontologyStatus”. The “ontologyStatus” provides the datatype property “inconsistent”. To have a hint about the reason for the inconsistency an object property “hasException” links from “ontology” to “exception”.

```

application(?x)  ^
filledOutByPerson(?x, ?y) ^
age(?y, ?z) ^
swrlb:lessThan(?z, 18) ^
ontology(?o)
  → hasException(?o, ExceptionPersonMustBeOlderThan18)

```

The SWRL rule above expresses that the ontology has the exception “ExceptionPersonMustBeOlderThan18”, if an application is filled out by a person who is younger than 18.

The atom “application(?x)” holds, if an instance of the concept “application” exists. The atom “filledOutByPerson” holds, if the instance of “application” is related to an instance of the concept “person” by property “filledOutByPerson”. If the found instance of “person” is of “age” ?z, this atom holds. The built-in relation “swrlb: lessThan” holds, if ?z is less than 18. If an instance of “ontology” is found, the condition of the last atom is fulfilled, so the whole condition is true. The consequence is that the instance “ExceptionPersonMustBeOlderThan18” of the concept “exception” is linked to the ontology by the object property “hasException”.

If a constraint is violated the status of the ontology is inconsistent. It can be expressed by following rule:

```

ontology(?x)  ^
hasException(?x, ?y) ^
ontologyStatus(?z) →
inconsistent(?z, true)

```

Guidelines

As with the constraints, guidelines can also be represented as logical rules, using the reserved property “warning” instead of inconsistent and the concept “guideline” to have a hint about the reason for the warning.

Subject	Predicate	Object
A confirmation	SHOULD BE sent	within 5 days

To express the guideline below the term “confirmation” should be represented as a concept. It contains the datatype-property “sent”.

```
confirmation(?x) ^
sent(?x, ?y) ^
swrlb:greaterThan(?y, 5) ^
ontology(?z)
  → violatedGuideline (?z, ConfirmationShouldBeSentWithin5Days ) ^
    accept(ConfirmationShouldBeSentWithin5Days, false)
```

The guideline has the datatype-property “accept”, so that an officer can accept the violation. To express that the status of the ontology is “warning”, the rule can be expressed as follows:

```
ontologyStatus(?x) ^
ontology(?y) ^
violatedGuideline(?y, ?z) ^
accept(?z, false)
  → warning(?y, true)
```

Action Enablers

Action enabling rules trigger planning and refinements of a process from predefined activities like the rule below.

	Term	Condition		Action
IF	a building	is closer than 50 meters to a natural water	DO	approve environmental compatibility

Based on the context of the actual service execution, action enabling rules associated to particular process steps are invoked at runtime to dynamically determine

and initiate the appropriate actions. Therefore, every process must be modeled as an OWL-S process. To express the sample action enabling rule above, an atomic process²³ called “ApproveEnvironmentalCompatibility” must be executed.

To present the condition of the rule an instance of the class “SWRL-Condition” of the OWL-S-Ontology must be created and the following condition must be entered at the property “expressionObject”.

```
building (?x) ^
distanceToNaturalWater (?x, ?y) ^
swrlb:lessThan(?y, 50)
```

To express the consequence, an instance of the class “If-Then-Else”, which is subclass of “ControlConstruct” has to be created, which object-property “ifCondition” link to the condition and the property “then” link to the atomic process.

Computations

The statement provides an algorithm to compute the value of a term.

Subject	Computation
Fee	IS COMPUTED AS days of delay x 10 CHF

To represents the computational rule, a concept “BuildingApplication” is used. This concept contains the two datatype properties “daysOfDelay” and “fee”. These two properties are used to compute the product.

```
BuildingApplication(?x) ^
daysOfDelay(?x, ?y)
-> swrlb:multiply(?fee, ?y, 10) ^
fee(?x, ?fee)
```

²³A process model is composed of atomic processes and/or composite processes. An atomic process is defined as a non-decomposable process, e.g. in a process implementation with web-services it can be executed using a single call.

For every instance of “BuildingApplication” which has a delay, the fee is computed as the value of “daysOfDelay” multiplied by 10. Similar to multiplication, built-ins for subtraction, addition and division can be used.

Negation

A problem exists in expressing negations. In the above examples negation is expressed by an inverse predicate “lt” (less than) instead of “NOT greater than”. However, in the current version of SWRL such inverse predicates are not available in every case. For example, the rule “If the construction plan is not available, do request documents.” cannot be represented directly in SWRL. This constraint can be expressed by a rule that stated that there is an inconsistency. This kind of negation is called negation as failure: it is true if no construction plan is mentioned in the knowledge base.

Rule Set

Every rule expressed in SWRL can be combined into rule sets. For example, all rules which are defined for a process to evaluate an application form can be combined in a rule set called “FormEvaluationRules”. Our approach is to use the following three concepts to express rule sets: “Rule”, “RuleSet” and “File” (Fig. 19.11).

The concept “Rule” provides the datatype properties “RuleID”, “RuleName” and “RuleDescription” to store the name, id and description of a rule. The SWRL-file itself will be stored in an external file. To retrieve the file, the instance of ‘File’ contains the datatype property “filePath”. To have a relation between a file and the expressed rule, the ontology provides the object properties “containsRule” and “isStoredIn” respectively. All rules could be combined into rulesets by creating an instance of “RuleSet” and using the object property “comprisesRule” to link to the rules. One benefit of not storing rules directly in a rule set is that a rule can be combined to multiple rule sets.

To have a link between processes and rule set, the concept “process” of the OWL-S ontology has to be extended by the object property “containsRuleSet”, the range of which links to “RuleSet”.

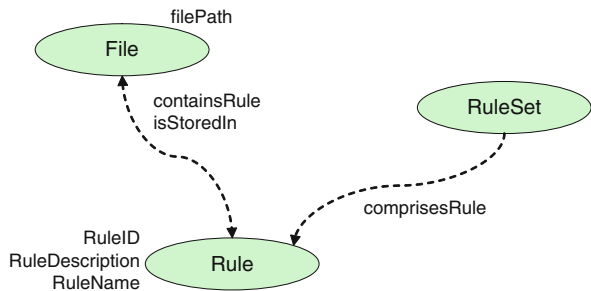


Fig. 19.11 Rule ontology

19.6 Ontologies for Modelling Agile E-Government Processes²⁴

Although there are legally binding rules and regulations every administration has to obey, dealing with people's concerns means dealing with different circumstances every time. In this sense, e-government services are often knowledge intensive processes, where the actual process execution and the involved participants and administrations depend on various factors. Consider for example the process of building permission in a European municipality. Under particular circumstances an environmental compatibility check is required or the historical preservation agency has to be involved, in which case complex clarifications have to be made. Modelling all possible variants of a process can lead to complex process models. Sometimes this modelling can even be impossible if the tasks are mutually dependent on one another.

To deal with these kind of agile processes, the Agile Process Management (APM) approach combines business and knowledge work processes by linking process models and business rules (Hinkelmann et al., 2006b). The business rules extend process execution on three ways:

- *Variable process execution*: Determine activities and processes to be executed thereby accounting for dependencies between activities
- *Intelligent resource allocation at run time*: Selection of employees based on special skills and selection of particular web services adequate for the actual circumstances
- *Intelligent branching and decision making*: deriving workflow-relevant data using inferences and computing values
- *Consistency checking*: Avoid violation of integrity constraints and guidelines

Agile processes management includes all of the previous modelling approaches to provide adaptable, flexible and reusable e-government services: *Ontologies* build the basis for modelling and executing *semantically enriched processes* and *business rules*.

At run time action-enabling rules select the activities that have to be executed depending on the actual context of the process instance. Inference rules allow for resource allocation and support the user in decision making, while integrity constraints and guidelines (in combination with inference rules) ensure consistency checking and compliance.

Figure 19.12 shows the three aspects of ontology modelling and use:

- domain modelling to make business knowledge explicit,
- business rules modelling to make rules and regulations explicit,

²⁴A process is considered “agile” when its execution model is created flexible at runtime, based on the results of triggered rules instead of static pre-defined models.

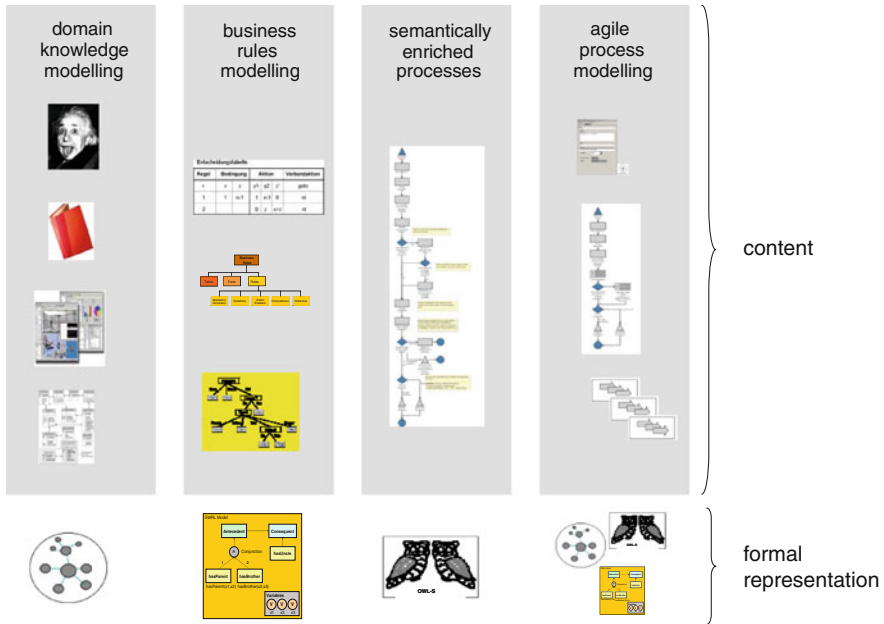


Fig. 19.12 Modelling agile services with ontologies, rules and semantically enriched process models

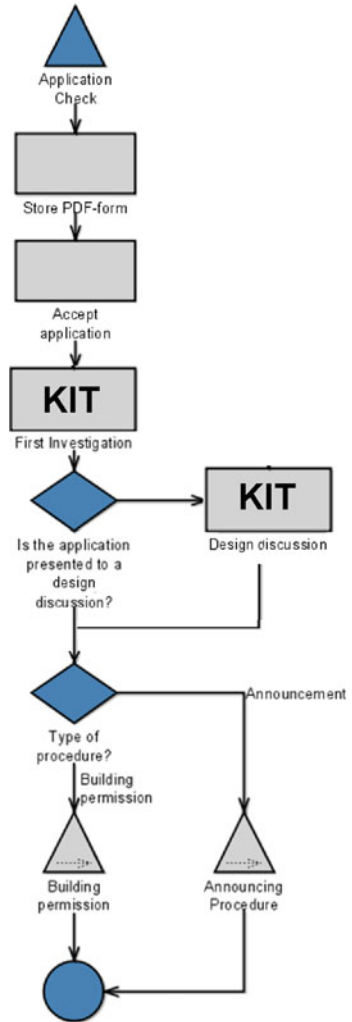
– process (life-cycle) metadata to make design decisions explicit and allow for change management.

The far right of the figure depicts the agile process modelling, combining ontologies with rules. Let us explain the agile process management framework with an example:

The first phase of designing agile processes is sketching a “process skeleton” that means the main activities (atomic processes) of a (composite) process. Figure 19.13 depicts a skeleton of a process model for building permission where parts are marked as knowledge intensive tasks. As already mentioned, depending on particular circumstances different activities are to be executed during the process, e.g. checking for environmental compatibility or historical preservation. The conditions for these activities are not always given at the beginning but may appear only after additional clarifications are made. Thus, we have an unforeseeable sequence of data collections, clarifications and decision making which cannot be modelled exactly but must be determined at run-time. Decision making, however, may require specific skills, in which case experts with relevant experience should be allocated to perform the task. Since laws and regulations in this area are quite complex, a lot of guidelines and constraints have to be considered.

Thus, the knowledge-intensive parts in the process can then be regarded as agile, containing variable processes, intelligent branching and flexible resource allocation. To cope with this agility, we extended process modelling in two ways:

Fig. 19.13 Building permission process skeleton with knowledge-intensive parts



- First, to each activity we can associate sets of rules that are executed at run time and affect the execution by selecting resources, deriving values and checking constraints and guidelines. As shown in Section 19.5, these rules refer to concepts modeled in an ontology.
- Second, we have a new modelling object for variable process parts. A variable process corresponds to a subprocess with the particularity that the activities of the subprocess are determined at run-time instead of modeling time using action-enabling rules (see Section 19.5.1).

In Fig 19.14 the activities with a question mark are variable processes. The first variable process has a reference to various possible activities: formal investigation,

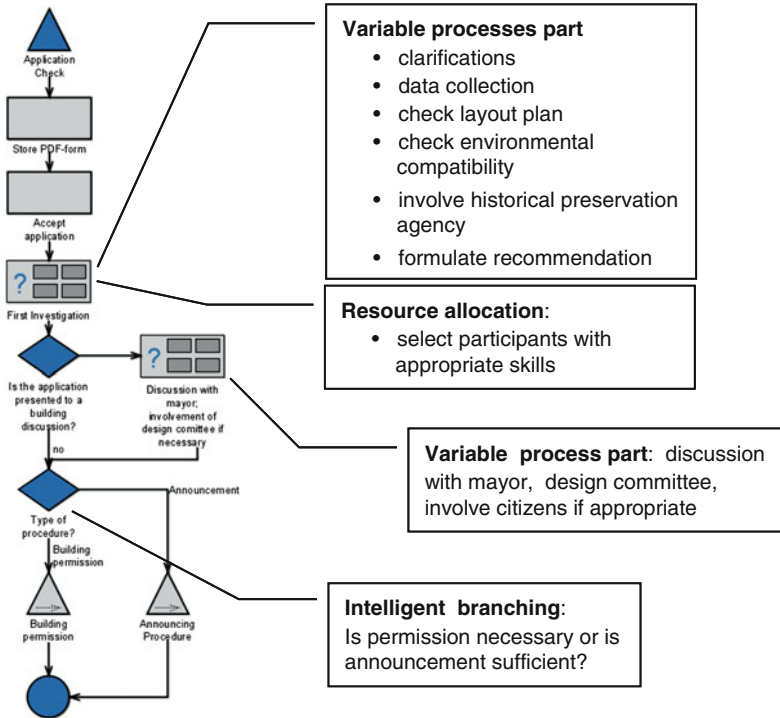


Fig. 19.14 Building permission as agile process

collection of data, check layout plan, check environmental compatibility, check for historical preservation, and formulate recommendation. The second one is executed if a design discussion with the mayor, experts and citizens is necessary.

Intelligent branching depends on inference and computation rules that are executed at run time. In the sample process there are two possibilities depending on the decision in previous phases: One branch starts a subprocess to handle a building permission application. The second branch leads to a subprocess for simpler cases where a building announcement is sufficient and no formal permission is necessary.

Using OWL, OWL-S and SWRL as interchange formats allows the migration of ontologies, process models and rules into executable code that can be Java, BPEL or commercial software to provide the appropriate interfaces. With that approach the agile process model can be executed. Fig. 19.15 shows the overall picture of agile process management. Rule sets are associated to business process models, terms and facts of the business rules are defined in an ontology. For execution a rule engine²⁵ is linked to a workflow engine,²⁶ both having access to application data.

²⁵A business rule engine or inference engine is a software component that separates the business rules from the application code and allows deriving answers from a knowledge base.

²⁶“The workflow enactment software interprets the process description and controls the instantiation of processes and sequencing of activities, adding work items to the user work lists and

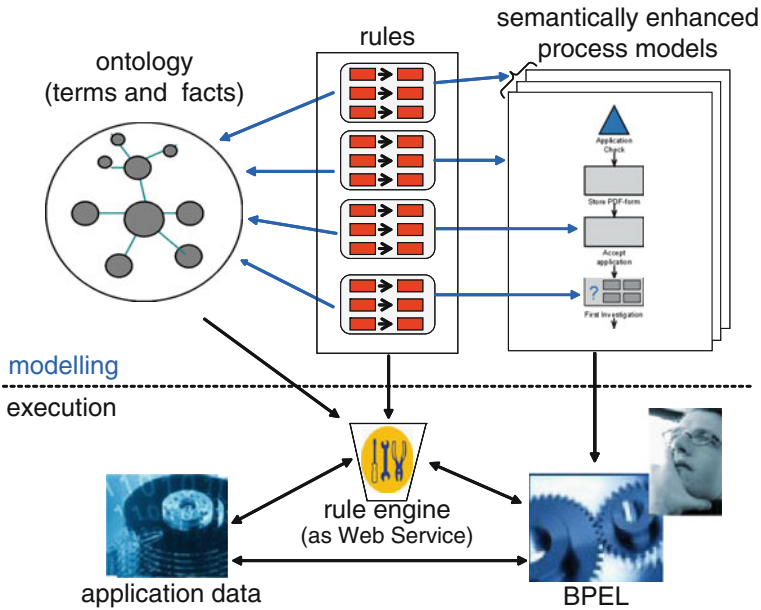


Fig. 19.15 Agile process management framework

19.7 Conclusion

Recent studies have shown that there is a significant advance in e-government in nearly every country but there are still many challenges. In this chapter we have presented some work to reinforce e-government that is based on the use of ontologies.

One of the greatest challenges is the effort required in building and maintaining ontologies. Although the public sector is strongly regulated and the services are nearly identical in many municipalities and public administrations, there is still no standard vocabulary. Even life events, a concept used to organise most of the portals and web sites differ in naming, granularity and structure. In many countries, however, there are projects and initiatives to define a standard vocabulary for describing resources and services. This will have an important influence in finding services on the web and in interoperability of services between public administrations. There are also international consultations to enforce compatibility, in particular in the European Union.

invoking application tools as necessary. This is done through one or more co-operating workflow management engines, which manage(s) the execution of individual instances of the various processes.” (TC00-1003 Issue 1.1 Workflow Reference, Workflow Management Coalition Page 14 of 14, URL: <http://www.wfmc.org/standards/docs/tc003v11.pdf>.)

We presented an approach for ontology building that starts with the collection and consolidation of terms which in subsequent steps are related to facts and then can be transformed to ontologies. A major advantage of this approach is that it takes into account the fact that domain experts in the public administrations must be involved in this process.

Building ontologies, however, is only an initial step. We also described how ontologies can be used for semantic process modelling. This allows a public administration to exchange experiences in process design and lifecycle management. In combination with business rules this semantically enhanced process modelling results in what we called agile process management.

References

- Abecker, A., A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. 1998. Toward a well-founded technology for organizational memories. *IEEE Intelligent Systems and their Applications* 13(3) S40–48, May/June 1998. Reprint in *The Knowledge Management Yearbook 1999–2000*, eds. J.W. Cortada, and J.A. Woods, 185–199. Boston: Butterworth-Heinemann.
- Borst, P. 1997. Construction of engineering ontologies for knowledge sharing and reuse, Phd thesis, Enschede, The Netherlands: University of Twente. <http://purl.org/utwente/fid/1392>. Retrieved, 15 Dec 2006.
- Business Rule Group. 2006. What is a business rule? <http://www.businessrulesgroup.org/defnbrg.shtml>. Retrieved, 05 Dec 2006.
- Capgemini Consulting. 2006. Online availability of public services: How is europe progressing? Web Based Survey on Electronic Public Services, Report of the 6th Measurement, June 2006. http://www.capgemini.com/resources/thought_leadership/2006_online_availability_of_public_services/. Retrieved, 22 Dec 2006.
- Chappel, O. 2006. Term–fact modeling, the key to successful rule-based systems. <http://www.brcommunity.com/b250.php>. Retrieved, 22 Nov 2006.
- Conklin, J., and M. Begeman. 1998. gIBIS – A hypertext tool for exploratory policy discussion. In *Proceedings of CSCW98*, 140–152. Seattle, WA.
- Daconta, M. C., L.J. Oberst, and K.T. Smith. 2003. *The semantic web*. New York, NY: Wiley.
- Dublin Core Metadata Initiative. 2006. DCMI metadata terms. <http://dublincore.org/documents/dcmi-terms/>. Retrieved, 23 Dec 2006.
- eCH. 2006. Best practice struktur prozessinventarliste. Standard Recommendation No. eCH-0015 of eCH eGovernment Standards. http://www.ech.ch/index.php?option=com_docman&task=doc_download&gid=326. Retrieved, 5 Jan 2007.
- eCH. 2008. eCH-0049 Vernehmlassungseingaben Themenkatalog Privatpersonen. Standard Recommendation No. eCH-0049 plus addenda of eCH eGovernment Standards. http://ech.ch/index.php?option=com_docman&task=cat_view&gid=118&Itemid=181&lang=de. Retrieved, 01 Sep 2008.
- Fernández-López, M., and A. Gómez-Pérez. 2003. Overview and analysis of methodologies for building ontologies. *Cambridge Journals* 17:129–156.
- Fernandez, M., A. Gomez-Perez, and N. Juristo. 1997. METHONTOLOGY: From ontological arts towards ontological engineering. In *Proceedings of the AAAI-97 Spring Symposium Series on Ontological Engineering*, 33–40. Stanford, CA.
- Fraser, J., N. Adams, A. Macintosh, A. McKay-Hubbard, T.P. Lobo, P.F. Pardo, R.C. Martínez, and J.S. Vallecillo. 2003. Knowledge management applied to egovernment services: The use of an ontology. *Knowledge Management in Electronic Government: 4th IFIP International Working Conference, KMGov 2003*, Rhodes, Greece, May 26–28, Proceedings. Berlin/Heidelberg: Springer.

- Gandits, F., and R. Schaffer. 2006. Standardattribute für Leistungen und Leistungsgruppen. Recommendation of the Austrian e-Government Bund-Länder-Gemeinden. <http://reference.e-government.gv.at/uploads/media/st-att-1-0-0-2004-0102.pdf>. Retrieved, 05 Jan 2007.
- Gómez-Pérez, A. 2001. Evaluating ontologies. <http://www3.interscience.wiley.com/cgi-bin/fulltext/77002631/PDFSTART>. Retrieved, 15 Dec 2006.
- Gruber, T.R. 1993a. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2):199–220.
- Gruber, T.R. 1993b. Toward principles for the design of ontologies used for knowledge sharing. In *Formal ontology in conceptual analysis and knowledge representation*, eds. N. Guarino, and R. Poli, The Netherlands: Kluwer Academic Publishers. For download at, <http://www2.umassd.edu/SWAagents/agentdocs/stanford/ontodesign.pdf>. Retrieved, 05 Jan 2007.
- Grüninger, M., and M.S. Fox. 1995. Methodology for the design and evaluation of ontologies. <http://www.eil.utoronto.ca/EIL/public/method.ps>. Retrieved, 15 Dec 2006.
- Gugliotta, A., V. Tanasescu, J. Domingue, R. Davies, L. Gutiérrez-Villarías, M. Rowlatt, M. Richardson, and S. Stinčić. 2006. Benefits and challenges of applying semantic web services in the e-Government domain. http://kmi.open.ac.uk/projects/dip/resources/Semantics2006/Semantics2006_DIP_Camera_Ready.pdf. Retrieved, 15 Dec 2006.
- Gugliotta, A. 2006. Knowledge modelling for service-oriented applications in the e-Government domain. <http://www.dimi.uniud.it/gugliott/thesisgugliotta.pdf>. Retrieved, 15 Dec 2006.
- Swiss Federal Strategy Unit for Information Technology FSUIT. 2004. HERMES – Management and execution of projects in information and communication technologies (ICT). http://www.hermes.admin.ch/ict_project_management/manualsutilities/manuals/hermes-foundations/at_download/file. Retrieved, 06 Jan 2007.
- Hinkelmann, K., F. Probst, and B. Thönssen. 2006a. Reference modeling and lifecycle management for e-Government services. In eds. A. Abecker, A. Sheth, G. Mentzas, and L. Stojanovic, *Semantic Web Meets e-Government, Papers from the 2006 AAAI Spring Symposium*, AAAI Technical Report SS-06-06.
- Hinkelmann, K., F. Probst, and B. Thönssen. 2006b. Agile process management framework and methodology. In eds. A. Abecker, A. Sheth, G. Mentzas, and L. Stojanovic, *Semantic Web Meets e-Government, Papers from the 2006 AAAI Spring Symposium*, AAAI Technical Report SS-06-06.
- Horrocks, I., P. Patel-Schneider, H. Boley, S. Tabet, B. Groszof, and M. Dean. 2004. SWRL: A semantic web rule language, combining OWL and RuleML. <http://www.w3.org/Submission/SWRL/>. Retrieved, 08 Dec 2006.
- Chevallerau, F.-X. 2005. eGovernment in the member states of the European Union, Brussels. <http://europa.eu.int/idabc/egovo>. Retrieved, 05 Jan 2007.
- Kalfoglou, Y., and M. Schorlemmer. 2005. Ontology mapping: The state of the art. <http://drops.dagstuhl.de/opus/volltexte/2005/40/pdf/04391.KalfoglouYannis.Paper.40.pdf>. Retrieved, 25 Dec 2006.
- Kunz, W., and H.W.J. Rittel. 1970. Issues as elements of information systems. WP-131, University of California. <http://www-iurd.ced.berkeley.edu/pub/WP-131.pdf>. Retrieved, 5 Dec 2005.
- Macias, J.A., and P. Castells. 2004. Providing end-user facilities to simplify ontology-driven web application authoring. *Interacting with Computers* 19(4):563–585, July 2007.
- Maedche, A., and S. Staab. 2004. Ontology learning. In *Handbook on ontologies*, eds. S. Staab, and R. Studer, 173–189. New York, NY: Springer.
- McGuinness, D.L., and F. van Harmelen, eds. 2004. OWL web ontology language overview: W3C Recommendation 10 Feb 2004. <http://www.w3.org/TR/owl-features/>. Retrieved, 15 Dec 2006.
- Noy, N.F., and D.L. McGuinness. 2002. Ontology development 101: A guide to creating your first ontology. http://protege.stanford.edu/publications/ontology_development/ontology101.pdf. Retrieved, 15 Dec 2006.
- Martin, D., ed. 2004. OWL-S: Semantic markup for web services. W3C Member Submission 22 Nov 2004. <http://www.w3.org/Submission/OWL-S/>. Retrieved, 06 Jan 2007.

- Peristeras, V., and K. Tarabanis. 2006. Reengineering public administration through semantic technologies and the GEA domain ontology. AAAI Spring Symposium on Semantic Web Meets e-Government, Stanford University, March 2006. <http://www.semantic-gov.org/index.php?name=UpDownload&req=getit&lid=1>. Retrieved, 15 Dec 2006.
- Pinto, H.S., and J.P. Martins. 2002. Evolving ontologies in distributed and dynamic settings. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, eds. D. Fensel, F. Giunchiglia, D.L. McGuinness, M.A. Williams, San Francisco, CA: Morgan Kaufmann.
- Pinto, H.S., and J.P. Martins. 2004. Ontologies: How can they be built? *Knowledge and Information Systems, Computer Science*, 6(4). <http://springerlink.metapress.com/content/0p5yqrdh5t5dvd06/fulltext.pdf>. Retrieved, 16 Dec 2006.
- Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. 1999. *Knowledge engineering and management: The CommonKADS methodology*. Cambridge, MA: MIT Press.
- Stojanovic, L., A. Abecker, D. Apostolou, G. Mentzas, and R. Studer. 2006. The role of semantics in eGov service model verification and evolution. AAAI Spring Symposium on Semantic Web Meets e-Government, Stanford University, March 2006.
- Sure, Y., M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. 2002. OntoEdit: Collaborative ontology development for the semantic web. In *International Semantic Web Conference ISWC 2002*, LNCS 2342, eds. I. Horrocks, and J. Hendler, 221–235. Heidelberg: Springer. http://www.aifb.unikarlsruhe.de/WBS/ysu/publications/2002_iswc_ontoedit.pdf. Retrieved, 15 Dec 2006.
- Uschold, M., M. King, S. Moralee, and Y. Zorgios. 1998. The enterprise ontology. *Cambridge Journals* 13:31–98.
- Uschold, M. 1996. Building ontologies: towards a unified methodology. *Proceedings of Expert Systems 96*, Cambridge, Dec 16–18 1996.
- Von Halle, B. 2002. *Business rules applied, building better systems using the business rules approach*, New York, NY: Wiley.
- Wimmer, M., and E. Tambouris. 2002. Online one-stop government: A working framework and requirements. In *Proceedings of the IFIP World Computer Congress*, Montreal, Aug 26–30 2002.
- A. Wroe. 2006. TERREGOV white paper: Semantic based support to civil servants. http://www.terregov.eupm.net/my_spip/index.php?param=12. Retrieved, 15 Dec 2006.

Chapter 20

An Ontology-Driven Approach and a Context Management Framework for Ubiquitous Computing Applications

Christos Goumopoulos and Achilles Kameas

20.1 Introduction

Pervasive or Ubiquitous computing is a new technological paradigm in which every thing around us has built-in and interconnected computers (Weiser, 1991; Disappearing Computer, 2007). Embedded in everyday objects these interconnected devices (also called artifacts) open up an unlimited number of possibilities for many new applications and services (Norman, 1999; Bergman, 2000). Applications result from the dynamic and adaptive composition of such artifacts, triggered via explicit user/application requests, application/task templates, or even more autonomic interaction schemes. Then, in this context, a “system” is defined to be the collective, complex service that emerges as an aggregation of simpler services offered by independent artifacts.

Ontologies can help address some key issues of Ubiquitous computing environments such as knowledge representation, semantic interoperability and service discovery. One important issue, for example, to be resolved in building a Ubiquitous computing system, is the development of interfaces between heterogeneous and incompatible components, objects or artifacts. This can be dealt with by developing an ontology of concepts so that different types of functionality and interactions or artifact bindings can be described in a way that is natural and consistent across different systems. If the services provided by artifacts are to be properly exploited, then it must be ensured that they will be able to interpret the representations sent to them and to generate the representations expected from them. Following a service-oriented approach, applications state their requirements in terms of concepts that are part of the application’s ontology rather than specific resource instances.

It is also important to capture complex interactions between many different artifacts. Thus, apart from simple peer-to-peer interactions, appropriate descriptions are needed to support more complex interaction structures; both synchronous

C. Goumopoulos (✉)
Distributed Ambient Information Systems Group, Hellenic Open University & Computer
Technology Institute, Patras, Hellas, Greece
e-mail: goumop@cti.gr

and asynchronous schemes are required to cater for the complexity of pervasive computing applications. To achieve synergy, the system has to apply global decision making procedures in order to cope with distributed resource management, service composition and performance optimization. At the same time, each artifact employs local decision making procedures in order to adjust its autonomous operation to changes of context, to learn and to maintain its stable operation.

The goal of this chapter is to present an ontology-driven approach and a context management framework for the composition of context-aware Ubiquitous computing applications. The next section outlines how context is modeled and used in various Ubiquitous computing systems emphasizing on ontology-oriented approaches. Then we describe the ontology that was developed in order to conceptually represent context-aware Ubiquitous computing systems. This ontology is designed taking into account both the autonomous nature of components, objects and artifacts and the necessity of their interoperability; so it is divided into two layers, a private (application-specific) and a common (core) one. The core ontology defines a meta-model of the Ubiquitous computing domain based on the Bunge-Wand-Weber (BWW) ontology. Then we present a hierarchical approach for engineering context-aware Ubiquitous computing systems including the context management and decision-making processes as well as the analysis of the mechanism that was developed based on that processes. Finally, we conclude by evaluating our ontology-driven approach and presenting the lessons learned. A prototype application is also outlined where an augmented plant is incorporated in a Ubiquitous computing environment in order to collaborate with other augmented objects, providing thus a communication channel between plants and people.

20.2 Ontology Based Modeling of Context Aware Ubiquitous Computing Systems

According to (Dey, 2001) context is: *“Any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computation and physical objects.”* In Ubiquitous computing applications different kinds of context can be used like physical (e.g. location and time), environmental (e.g. weather and light) and personal information (e.g. mood and activity). Nevertheless, the term context mostly refers to information relative to location, time, identity and spatial relationships.

A number of informal and formal context models have been proposed in various Ubiquitous computing systems; a survey of context models is presented in (Strang and Linnhoff-Popien, 2004). Among systems with informal context models, the Context Toolkit (Dey et al., 2001) represents context in form of attribute-value tuples. The Cooltown project (Kindberg et al., 2000) proposed a Web based model of context in which each object has a corresponding Web description. Both ER and UML models are used for the representation of formal context models by (Henricksen et al., 2002).

As ontologies are a promising instrument to specify concepts and their interrelations, they can provide a uniform way for specifying the core concepts of a context model, as well as an arbitrary amount of subconcepts and facts, altogether enabling contextual knowledge sharing and reuse in a Ubiquitous computing system (De Bruijn, 2003). Thus several research groups have presented ontology-based models of context and used them in Ubiquitous computing applications. In the following, we briefly describe the most representative ones.

In the GAIA framework (Ranganathan and Campbell, 2003), an infrastructure is presented that supports the gathering of context information from different sensors and the delivery of appropriate context information to Ubiquitous computing applications. The project aim was to develop a flexible and expressive model for context able to represent the wide variety of possible contexts and to support complex reasoning on contexts. Context is represented with first-order predicates written in DAML+OIL. This context model allows deriving new context descriptions from other sensed context.

GLOSS (GLObal Smart Space) is a software infrastructure that enables the interactions between people, artifacts, and places, while taking account of both context and movement on a global scale (Dearle et al., 2003). By exploiting the features of physical spaces, it uses people's location and movement as a source of task-level context and as a guide to provide appropriate information, or services. Therefore, GLOSS facilitates the low-level interactions (such as tracking a user's location) that are driven by high-level contexts (such as a user's task). This system accommodates both service heterogeneity and evolution, using ontologies. The GLOSS ontologies describe concepts, which provide the precise understanding of how services (physical and informational) are used and how users interleave various contexts at run time.

CoBra (Context Broker Architecture) is a pervasive context-aware computing infrastructure that enables Ubiquitous agents, services and devices, to behave intelligently according to their situational contexts (Kagal et al., 2001). It is a broker-centric agent architecture that provides knowledge sharing, context reasoning, and privacy protection support for Ubiquitous context-aware systems, using a collection of ontologies, called COBRA-ONT, for modeling the context in an intelligent meeting room environment (Chen et al., 2003). These ontologies are expressed in the Web Ontology Language (OWL), define typical concepts associated with places, agents, and events and are mapped to the foundational ontologies that are relevant to smart spaces.

Wang et al. created an upper ontology, the CONON (Wang et al., 2004) context ontology, which captures general features of basic contextual entities, a collection of domain specific ontologies and their features in each subdomain. The upper ontology is a high-level ontology which defines basic concepts about the physical world such as "person", "location", "computing entity", and "activity". The domain-specific ontologies, are a collection of low-level ontologies, which define the details of general concepts and their properties in each sub-domain where they apply to (like home domain, office domain). All these context ontologies help in sharing a common understanding of the structure of contextual information coming from users, devices, and services, so as to support semantic interoperability and

reuse of domain knowledge. The CONON ontologies are serialized in OWL-DL which has a semantic equivalence to the well researched description logic (DL). Thus CONON supports two types of reasoning: reasoning to detect and correct inconsistent context information and reasoning as a means to derive higher level context information. The latter type of reasoning is based on properties like symmetry and transitivity, as well as on user-defined rules. The CONON ontology is part of the SOCAM (Service-Oriented Context-Aware Middleware) architecture, which supports the building and rapid prototyping of context-aware services in pervasive computing environments (Gu et al., 2004).

The Context Ontology Language (CoOL) (Strang et al., 2003) is based on the Aspect-Scale-Context Information (ASC) model. Aspects represent classifications (e.g. Temperature), while scales are individual dimensions of aspects (e.g. Celsius). Context information is attached to a particular aspect and scale; quality metadata (e.g. meanError) is associated with information via quality properties. This contextual knowledge is evaluated using ontology reasoners, like F-Logic and OntoBroker. In addition to the determination of service interoperability in terms of contextual compatibility and substitutability, this language is used to support context-awareness in distributed service frameworks for various applications.

The CADO (Context-aware Applications with Distributed Ontologies) framework (De Paoli and Loregian, 2006) relies on distributed ontologies that are shared and managed in a peer-to-peer fashion. It is composed of three layers and designed to support mobility of workers in complex work settings. The three layers ensure semantic interoperability via the process of ontology merging, while context and application interoperability are ensured using Context and Interaction Managers respectively.

The CoCA (Collaborative Context-Aware) system is a collaborative, domain independent, context-aware middleware platform, which can be used for context-aware application development in Ubiquitous computing (Ejigu et al., 2007). This architecture for context-aware services focuses on context-based reasoning in Ubiquitous computing environments, using semantic-based collaborative approaches. The model uses an ontology for modeling and management of context semantics and a relational database schema for modeling and management of context data. These two elements are linked through the semantic relations built in the ontology.

The GAS Ontology (Christopoulou and Kameas, 2005) is based on a different approach for modeling Ubiquitous computing applications that are composed of artifacts. It adopts GAS, the Gadgetware Architectural Style (Kameas et al., 2003) according to which artifacts are called eGadgets, their services are called Plugs and the combination of two services is called a Synapse. GAS Ontology aims to conceptualize GAS by describing the semantics of these basic terms and by defining the relations among them. Thus the GAS Ontology provides a shared means for the communication and collaboration among artifacts, even though they may be produced by different manufacturers. This approach serves as the basis of our work that is presented later in this chapter.

Although each research group follows a different approach for using ontologies in modeling and managing context in Ubiquitous computing applications, it has been acknowledged by the majority of researchers (Dey, 2001; Henriksen et al., 2002; Ranganathan and Campbell, 2003; Christopoulou and Kameas, 2005) that it is a necessity to decouple the process of context acquisition and interpretation from its actual use. This can be achieved by introducing a consistent, reliable and efficient context framework which can facilitate the development of context-aware applications. In this respect, we propose an approach for a context-aware Ubiquitous computing system that eases the composition of such context-aware Ubiquitous computing applications and separates this process from the process of context acquisition.

The use of ontologies to model context-aware systems facilitates knowledge sharing across different systems and context reasoning based on semantic Web technologies. An important distinction between the approaches presented above and the one we adopted to develop the GAS ontology is that the former are based on understanding of ontology as a specification of some conceptualization (Guizzardi et al., 2002), whereas we approach ontology in philosophical terms, e.g. as in (Milton and Kazmierczak, 2004); this led to the development of an abstract meta-model for the Ubiquitous computing environment.

20.3 An Ontology-Driven Meta-Model for Ubiquitous Computing Systems

20.3.1 *Underlying Concepts*

From the system engineering perspective, conceptual modeling is at the core of systems analysis and design. Our approach for developing a conceptual model that represents the structural, relational and behavioral elements of the targeted Ubiquitous computing systems is based on the so-called Bunge-Wand-Weber (BWW) ontology. Ontology in this context represents a well-established theoretical domain within philosophy dealing with the models of reality. Wand and Weber have taken and extended an ontology presented by Mario Bunge (Bunge, 1977, 1979) and developed a formal foundation for modeling information systems (Wand and Weber, 1990). BWW Ontology has been widely applied in the information systems research field in contexts such as comparison of information systems analysis and design grammars, ontological evaluation of modeling grammars, information systems interoperability and for requirements engineering for commercial-off-the-shelf software and alignment in enterprise systems implementations (Rosemann et al., 2004).

Although the BWW ontology constructs have been originally defined using a rigorous set-theoretic language in many subsequent works the researchers attempted to simplify and clarify the explanation of the constructs by defining those using plain English (Weber, 1997). Following is the description of selected core ontological constructs of the BWW ontology:

- *Thing*: A thing is the basic construct in the BWV ontological model. The world is made of things that have properties. Two or more things (composite or primitive) can be associated into a *composite* thing.
- *Property*: We know about things in the world via their properties. A property is modeled via a function that maps the thing into some value. Properties are classified in a number of categories: *hereditary*, *emergent*, *intrinsic*, *binding/non-binding* and *mutual*.
- *Mutual Property*: A property that is meaningful only in the context of two or more things.
- *State*: The vector of values for all property functions of a thing is the state of the thing.
- *Conceivable State*: The set of all states that the thing may ever assume.
- *Stable state*: A stable state is a state in which a thing, subsystem, or system will remain unless forced to change by virtue of the action of a thing in the environment (an external event).
- *Transformation of a Thing*: A mapping from a domain comprising states to a co-domain comprising states.
- *System*: A set of things will be called a system, if, for any bi-partitioning of the set, interactions exist among things in any two subsets.
- *System Composition*: The things in the system are its composition.
- *System Environment*: Things that are not in the system, but which interact with things in the system are called the environment of the system.
- *System structure*: The set of couplings that exist among things within the system, and among things in the environment of the system and things in the system.
- *Subsystem*: A system whose composition and structure are subsets of the composition and structure of another system.

For developing a conceptual model for the Ubiquitous computing application domain, instead of using the entire BWV ontology, a more focused ontology is derived, by taking into consideration the requirements of the target application domain. Therefore, an appropriate subset of concepts is selected by applying elimination and specialization processes. In a similar perspective (Rosemann and Green, 2000), argue that taking into account the objectives of the modeling tasks in a specific problem domain as well as the type of users to be involved can assist in developing new ontologically based specific modeling grammars. In the next section we extend the concept of Thing to the concept of Entity and the concept of Composite Thing to the concept of Ambient Ecology. New concepts are introduced like the Plug and Synapse in order to provide detailed representation of the interaction among entities. The main advantage of this process is that the focused ontology is based on a well-established ontology with theoretical foundations. In addition, in order to communicate clearly and relatively easily the concepts of the derived conceptual model, we developed a description of the ontological constructs using a meta-model. Through this meta-model, the understanding of the ontological constructs and how they relate to each other can be explained clearly. We have used the UML meta-language for that purpose.

20.3.2 Focused Ontology

Our model defines the logical elements necessary to support a variety of applications in Ubiquitous computing environments. Its basic definitions are given below. A graphical representation of the concepts and the relations between them is given as a UML class diagram in Fig. 20.1.

eEntity: An eEntity is the programmatic bearer of an entity (i.e. a person, place, object, biological being or a composition of them). An eEntity constitutes the basic component of an Ambient Ecology. “e” stands here for extrovert. Extroversion is a central dimension of human personality, but in our case the term is borrowed to denote the acquired through technology competence of an entity to interact with other entities in an augmented way for the purpose of supporting the users’ everyday activities meaningfully. This interaction is mainly related with either the provision or consumption of context and services between the participating entities. A coffee maker, for instance, publishes its service to boil coffee, while context for a person may denote her activity and location. An augmented interaction between the coffee maker and the person is the activation of the coffee machine when the person awakes in the morning. For this to happen we need probably a bed instrumented with pressure sensors (an artifact) and a reasoning function for the persons’ awaking activity, which may not be trivial to describe. An eEntity in general possesses properties of three types: *structural* which belong to the entity itself; *relational* which relate the entity to other entities; and *behavioral* which determine possible changes to the values of structural and relational properties.

Artifacts: An artifact is a tangible object (biological elements like plants and animals are also possible) which bears digitally expressed properties; usually it is an object or device augmented with sensors, actuators, processing, networking unit etc. or a computational device that already has embedded some of the required

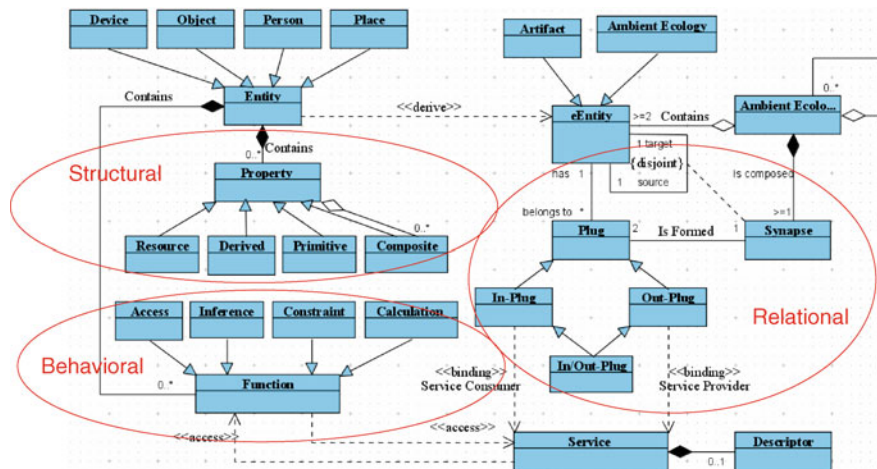


Fig. 20.1 A meta-model for the Ubiquitous computing Environment

hardware components. Software applications running on computational devices are also excessively considered to be artifacts. Examples of artifacts are furniture, clothes, air conditioners, coffee makers, a software digital clock, a software music player, a plant, etc.

Ambient Ecology: Two or more eEntities can be combined in an eEntity synthesis. Such syntheses are the programmatic bearers of Ambient Ecologies and can be regarded as service compositions; their realization can be assisted by end-user tools. Since the same eEntity may participate in many Ambient Ecologies the whole-part relationship is not exclusive. In the UML class diagram (see Fig. 20.1) this is implied by using the aggregation symbol (hollow diamond) instead of the composition symbol (filled diamond). Ambient Ecologies are synthesizable, because an Ambient Ecology is itself regarded as an eEntity and can participate in another Ambient Ecology.

Properties: eEntities have properties, which collectively represent their physical characteristics, capabilities and services. A property is modeled as a function that either evaluates an entity's state variable into a single value or triggers a reaction, typically involving an actuator. Some properties (i.e. physical characteristics, unique identifier) are entity-specific, while others (i.e. services) may be not. For example, attributes like color/shape/weight represent properties that all physical objects possess. The service light may be offered by different objects. A property of an entity composition is called an *emergent* property. All of the entity's properties are encapsulated in a *property schema* which can be send on request to other entities, or tools (e.g. during an entity discovery).

Functional Schemas: An entity is modeled in terms of a functional schema: $F = \{f_1, f_2, \dots, f_n\}$, where each function f_i gives the value of an observed property i in time t . Functions in a functional schema can be as simple or complex is required to define the property. They may range from single sensor readings to rule-based formulas involving multiple properties, to first-order logic so that we can quantify over sets of artifacts and their properties.

State: The values for all property functions of an entity at a given time represent the state of the entity. For an entity E , the set $P(E) = \{(p_1, p_2 \dots p_n) | p_i = f_i(t)\}$ represents the state space of the entity. Each member of the state vector represents a *state variable*. The concept of state is useful for reasoning about how things may change. Restrictions on the value domain of a state variable are then possible.

Services: Services are resources capable of performing tasks that form a coherent functionality from the point of view of provider entities and requester entities. Services are self-contained, can be discovered and are accessible through synapses. Any functionality expressed by a service descriptor (a signature and accessor interface that describes what the service offers, requires and how it can be accessed) is available within the service itself and is manifested by plugs.

Transformation: A transformation is a transition from one state to another. A transformation happens either as a result of an internal event (i.e. a change in the state of a sensor) or after a change in the entities' functional context (as it is propagated through the synapses of the entity).

Plugs: Plugs represent the interface of an entity. An interface consists of a set of operations that an entity needs to access in its surrounding environment and a set of operations that the surrounding environment can access on the given entity. Thus, plugs are characterized by their direction and data type. Plugs may be output (O) in case they manifest their corresponding property (e.g. as a provided service), input (I) in case they associate their property with data from other artifacts (e.g. as service consumers), or I/O when both happens. Plugs also have a certain data type, which can be either a semantically primitive one (e.g. integer, boolean, etc.), or a semantically rich one (e.g. image, sound etc.). From the user's perspective, plugs make visible the entities' properties, capabilities and services to people and to other entities.

Synapses: Synapses are associations between two compatible plugs. In practice, synapses relate the functional schemas of two different entities. Whenever the value of a property of a source entity changes, the new value is propagated to the target entity, through the synapse. The initial change of value caused by a state transition of the source entity causes finally a state transition to the target entity. In that way, synapses are a realization of the functional context of the entity.

20.3.3 Core vs. Application Ontology

The ontology that supports the development of Ubiquitous computing applications is divided in two basic layers: the Core and the Application layer. The discussed approach is in line with the design criteria proposed in (Gruber, 1993) for efficient development of ontologies:

- *Maximum monotonic extensibility:* new general or specialized terms can be included in the ontology in such a way that it does not require the revision of existing definitions.
- *Clarity:* terms which are not similar (common-sense terms vs. specialized domain ontologies) are placed in different taxonomies.

Core ontology – represents core knowledge of the Ubiquitous computing environment. This layer is designed to ensure *syntactic* and *structural* interoperability between various artifacts. Since the core ontology describes the language that artifacts use to communicate and collaborate it must be the same for all artifacts. A key issue is that the core ontology cannot be changed and contains only the necessary information in order to be small. In that way even artifacts with limited memory capacity can store and have access to the basic definitions. A graphical representation of the core ontology is given in Fig. 20.1. The basic classes of the core ontology have been discussed in the previous section.

Application ontology – represents knowledge about the application environments such as specific type of users and artifacts and acquired knowledge through

synapses. This layer is designed to ensure *semantic* interoperability. The knowledge represented by application ontology is described as instances of the classes defined in the core ontology. In that sense the application ontology is not a stand-alone ontology as it does not contain the definition of its concepts and their relations. The application ontology represents the description of each artifact that is related with an application containing information about physical properties, plugs and the services that are provided through these plugs. For example, the application ontology of the *eLamp* artifact contains knowledge about the physical characteristics of *eLamp*, such as luminosity, the description of a plug with an identifier “OnOff” based on the definition provided by core ontology as well as the declaration that this plug is associated with the service “light”. As services are the primary constituents of Ubiquitous computing systems the application ontology must contain specific service descriptions in order to support the service discovery and invocation mechanism. The application ontology describes also the synapses, the plugs of the artifact that participate in synapses, and the information about the capabilities/services of other artifacts that has been acquired via the synapses. Contrary to core ontology, the size of which must be small, the size of the application ontology can be as large as required, bounded only by the artifacts’ memory capacity. In addition, the application ontology is dynamic and can change over time without causing problems to artifact collaboration. The dynamic nature of the application ontology results from the knowledge that can be acquired through the various synapses that may be established between artifacts.

20.4 Context Management Framework

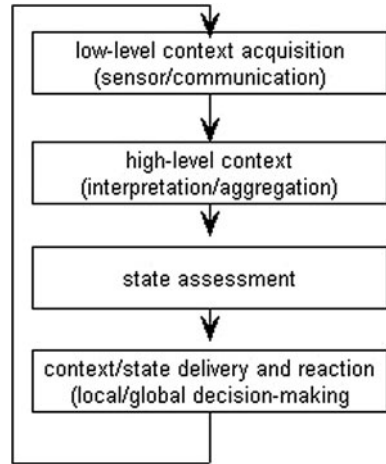
20.4.1 Context Management Process

A Ubiquitous computing application typically consists of an infrastructure used to capture context and a set of rules governing how the application should respond to changes in this context. In order to isolate the user from the process of context acquisition and management and on the other hand provide her with a Ubiquitous computing system that enables the composition of context-aware applications we propose that the system is organized in a hierarchy of levels.

The design approach for composing context-aware Ubiquitous computing applications needs to be backed by an engineering methodology that defines the correct formulation of the context and behavior. The proposed context management process is depicted in Fig. 20.2. The motivation for this process emerged from the fact that artifacts in Ubiquitous computing environment may be in different “states” that change according to the artifacts’ use by users and their reaction is based both on users’ desires and these states.

The first step in this context management process is the acquisition of the low level context, which is the raw data given by the sensors (Lexical Level). A set of sensors are attached to an artifact so that to measure various artifact parameters, e.g. the position and the weight of an object placed on an augmented table. As the output

Fig. 20.2 Context management process



of different sensors that measure the same artifact parameter may differ, e.g. sensors may use different metric system, it is necessary to interpret the sensors' output to higher level context information (Syntactical/Representation Level). Aggregation of context is also possible meaning that semantically richer information may be derived based on the fusion of several measurements that come from different homogeneous or heterogeneous sensors. For example, in order to determine if an object is placed on a table requires monitoring the output of table's position and weight sensors.

Having acquired the necessary context we are in a position to assess an artifact state (Reasoning Level) and decide appropriate response activation (Planning Level). Adopting the definition from Artificial Intelligence, a state is a logical proposition defined over a set of context measurements (Russell and Norvig, 2003). This state assessment is based on a set of rules defined by the artifact developer. The reaction may be as simple as turn on an mp3 player or send an SMS to the user, or it may be a complex one such as the request of a specific service, e.g. a light service. Such a decision may be based on local context or may require context from external sources as well, e.g. environmental context, location, time, other artifacts. The low (sensor) and high (fused) level data, their interpretation and the local and global decision-making rules are encoded in the application ontology. The basic goal of this ontology is to support a context management process that is based on a set of rules which determine the way that a decision is taken and must be applied on existing knowledge represented by this ontology. The description of the different types of these rules is given in the next section.

20.4.2 Rules

The application model specifies the behavior of an application and in our case this behavior is represented by Event-Condition-Action (ECA) rules. The categories

of rules that will support the decision-making process in the context management framework are as follows.

20.4.2.1 Rules for Artifact State Assessment

The left part of these rules denotes the parameters that affect the state of an artifact and the thresholds or the values for these specific parameters that lead to the activation of the rule, while the right part of these rules denotes the artifact state that is activated. These rules support the “translation” of low level context (values of parameters measured by sensors) to state assessment; they may also incorporate the translation from low level context to high level context (e.g. perform a set of operations to values measured by sensors like estimate the average value).

20.4.2.2 Rules for the Local Decision-Making Process

These rules exploit exclusively knowledge from the artifact that uses them. Their left part denotes the artifact states that must be detected and their possible activation level and their right part denotes the artifact requests and needs. When an artifact has a specific need we can consider that it needs a type of service offered by another artifact. When a rule from this category is activated, the artifact has to search its synapses in order to find a synapse which is associated to another artifact plug that provides the requested service. If such a synapse is found then the artifact can select it in order to satisfy its need. The situations, where more than one synapse is found that may be used to satisfy the request or no synapses are found, are handled by the local decision process using another kind of rules. The rules that define the final reaction of an artifact can be defined by the user or can be based on specifically user-defined policies. These rules support both the context delivery and the reaction of an artifact based on the local decision from state assessments.

20.4.2.3 Rules for the Global Decision-Making Process

These rules are similar to the rules for the local decision-making. Their main difference is that the rules for the global decision-making process have to take into account the states of other artifacts and their possible reactions so that to preserve a global state defined by the user.

20.4.3 Implementation

The architecture of the system that implements the aforementioned context management and reasoning process is shown in Fig. 20.3. The core modules of this system, Ontology Manager, Rule Manager and Inference Engine, are part of the updated version of the GAS-OS kernel (Drossos et al., 2007).

The Ontology Manager is the module responsible for the manipulation of knowledge represented into the artifact ontology. Specifically, it can only query the artifact

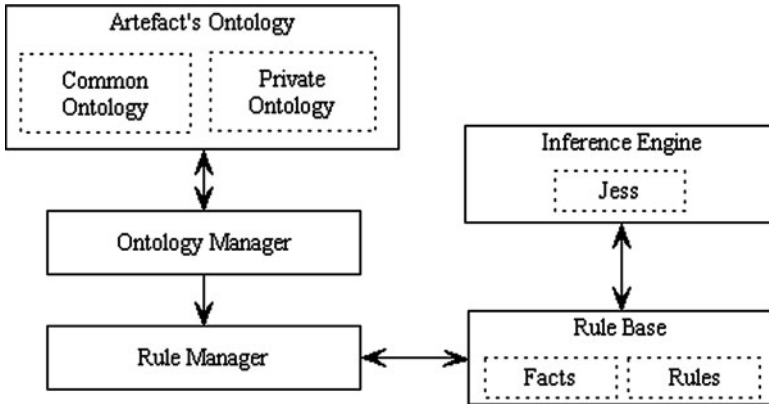


Fig. 20.3 Systems' architecture

common (core) ontology, since this ontology cannot be changed during the deployment of an application. On the other hand, it can both query and update the artifact private (application) ontology. The basic functionality of the Ontology Manager is to provide the other modules of the system with knowledge stored in the artifact ontology by adding a level of abstraction between them and the ontology.

The Rule Manager manages the artifact rule base and is responsible for both querying and updating this rule base. Note that the rules stored in an artifacts' rule base may only contain parameters, states and structural properties that are defined into the artifacts' private ontology. For the initialization of the context management process, apart from the rules, a set of initial facts are necessary. The Rule Manager is also responsible for the creation of a file containing the initial facts for an artifact. For example an initial fact may define the existence of an artifact by denoting its parameters, states and reactions that can participate in its rules and their initial values. In order to create such an initial fact, the Rule Manager uses knowledge stored in the artifacts' ontology. Subsequently, it queries the Ontology Manager for any information that it needs, like the artifacts' parameters, states and reactions.

The Inference Engine supports the decision-making process and is based on the Jess rule engine (Java Expert System Shell) (Jess, 2007). In order to initialize its process execution, the Inference Engine needs the artifact initial facts, which are defined by the Rule Manager, and the rules stored in the rule base. Note that in the current version of our system the rules in the rule base are stored in Jess format. The Inference Engine is informed of all the changes of parameters values measured by artifacts sensors. When it is informed of such a change, it runs all the rules of the rule base. If a rule is activated, this module informs the artifacts operating system of the activation of this rule and the knowledge that is inferred. The artifact's state and reaction is determined from this inferred knowledge.

The ontology describes the semantics of the basic terms of our model for Ubiquitous computing applications and their interrelations. One of the ontology goals is to describe the services that artifacts provide in order to support a service

discovery mechanism. Thus the Ontology Manager provides methods that query the ontology for the services that an artifact offers as well as for artifacts that provide specific services. GAS-OS gets from the Ontology Manager the necessary service descriptions stored in the artifact local ontology, in order to implement a service discovery mechanism. Finally the Ontology Manager using this mechanism and a service classification can identify artifacts that offer semantically similar services and propose objects that can replace damaged ones. Therefore, it supports the deployment of adaptive and fault-tolerant Ubiquitous computing applications.

20.4.4 Engineering Applications

To achieve the desired collective functionality a user has to form synapses by associating compatible plugs, thus composing applications using eEntities as components. The idea of building Ubiquitous computing applications out of components is feasible only in the context of a supporting component framework that acts as a middleware. The kernel of such a middleware is designed to support basic functionality such as accepting and dispatching messages, managing local hardware resources (sensors/actuators), the plug/synapse interoperability and a semantic service discovery protocol.

In terms of the application developer, plugs can be considered as context-providers that offer high-level abstractions for accessing context (e.g. location, state, activity, etc.). For example, an *eLamp* may have a plug that outputs whether the *eLamp* is switched on or switched off and an *eRoom* a plug informing if someone is in this room or not. In terms of the service infrastructure (middleware), they comprise reusable building blocks for context rendering that can be used or ignored depending on the application needs. Each context-provider component reads input sensor data related to the specific application and can output either low level context information such as location, time, light level, temperature, proximity, motion, blood pressure or high-level context information such as activity, environment and mood. An artifact has two different levels of context; the low level which contains information acquired from its own sensors and the high level that is an interpretation of its low level context information based on its own experience and use. Additionally an artifact can get context information from the plugs of other artifacts; this context can be considered as information coming from a “third-person experience”.

The application developers may establish synapses between plugs to denote both their preferences and needs and to define the behavior of the Ubiquitous computing application. From the service infrastructure perspective, the synapses determine the context of operation for each artifact; thus each artifact’s functionality is adapted to the Ubiquitous computing application’s structure.

By providing users with this conceptual model, we manage to decouple the low-level context management from the application business logic, which is captured as expressions in terms of high-level concepts that are matched with services available in the current application context. Instead of the classical approach of using established interfaces for resource access, this approach decouples the high-level concepts from the instances implemented by each context.

20.5 Prototype Application Example

20.5.1 Scenario

The example illustrated in this section deals with establishing communication between plants and artifacts. The prototype is a Ubiquitous computing application that is deployed in an indoor environment (e.g. home, office) and aims at facilitating the user in looking after her indoor plants. This ambient intelligence scenario demonstrates the concept of “communicating plant” (Goumopoulos et al., 2004) in the context of an every-day environment with several layers of decision-making.

The scenario is quite simple. A person has a plant in her office. However busy she may be, she still loves to take care of the plant. Several everyday objects are at her disposal for giving her an appropriate feedback (Lamp, MP3Player, Mobile Phone). Our aim is to form such an application where the plant will be able to provide the human with the appropriate feedback about its condition. The sequence of the scenario’s interactions is shown in Fig. 20.4.

The core artifact is the *ePlant*, which is constructed by adding sensors to the soil or the leaves of a plant. The *ePlant* “decides” whether it needs water or not by using its sensor readings and the appropriate decision making mechanism incorporated in it. The *eCarpet* is used to record whether the plant owner is inside her office or has stepped outside. Similarly, a *eMoodCube* (a glass brick that the user can set in

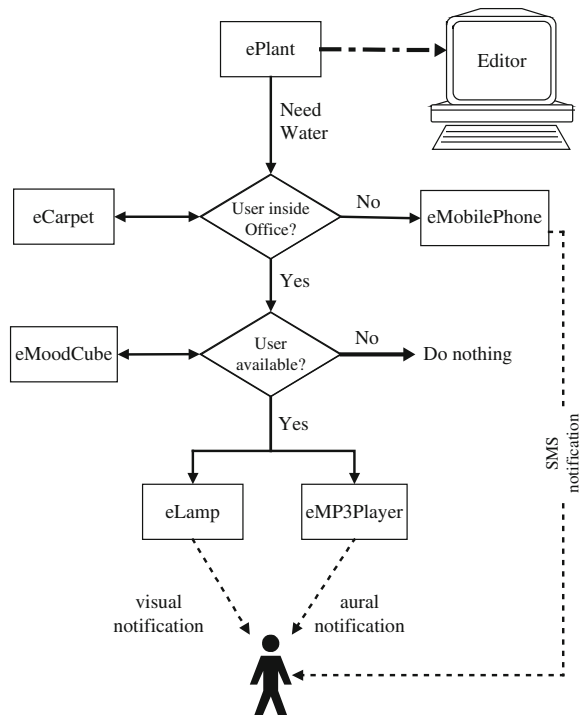


Fig. 20.4 Smart plant flowchart diagram

one of six possible positions) is used to provide an indication whether the user is available or not.

An augmented Lamp (*eLamp*) and an MP3Player (*eMP3Player*) are used to provide visual and aural notification respectively to the user. This notification is given according to the status of the *eMoodCube*. If the *eMoodCube* is set to “Do not disturb” status (which has been mapped by the plant owner to one of the six possible positions), the user is not notified at all. Lastly, in the case the user is not in her office, the application uses the *eMobilePhone* (an augmented Mobile Phone) to send her a SMS and inform her about the watering needs of the plant.

20.5.2 Components

The artifacts that are necessary for the realization of the above scenario are described as follows:

ePlant: The *ePlant* is capable of deciding whether it needs water or not, based on its sensor readings. These sensors fall into two categories: *Thermistors*, that is sensors that can perceive the temperature of the plants leaves and the environment and *Soil Moisture Probes*, which are able to measure the moisture level of the soil. Decision making rules, specific to the plant species, which combine the information given from the sensors above in order to provide a concrete decision on the current plant’s state, are added in the *ePlant* local ontology.

eMobilePhone: The *eMobilePhone* is a personal java enabled mobile phone, used for sending SMS to other mobile phones. When it receives a request to notify the user via an SMS, it will send the SMS to the corresponding telephone number.

eLamp: The *eLamp* is an augmented floor lamp used for visual notifications. The lamp switch is controlled by a micro-controller, which adjusts the state of the *eLamp* accordingly.

eCarpet: The *eCarpet* is an augmented carpet with pressure sensors attached, used for sensing the occupancy of the office (i.e. if the user is in the office). Based on the sequence that the pressure sensors are pressed, the *eCarpet* is capable of deducing if someone is entering or leaving a room, thus if the user is present or not.

eMoodCube: The *eMoodCube* is a glass brick containing a set of colored light bulbs with tilt switches attached. Each of the six different positions can be used for defining the current status or mood of the user.

eMP3Player: The *eMP3Player* is used to play audio files.

20.5.3 Implementation

A high-level view of the plugs and synapses that are necessary for the implementation of the Smart Plant application is given in Fig. 20.5.

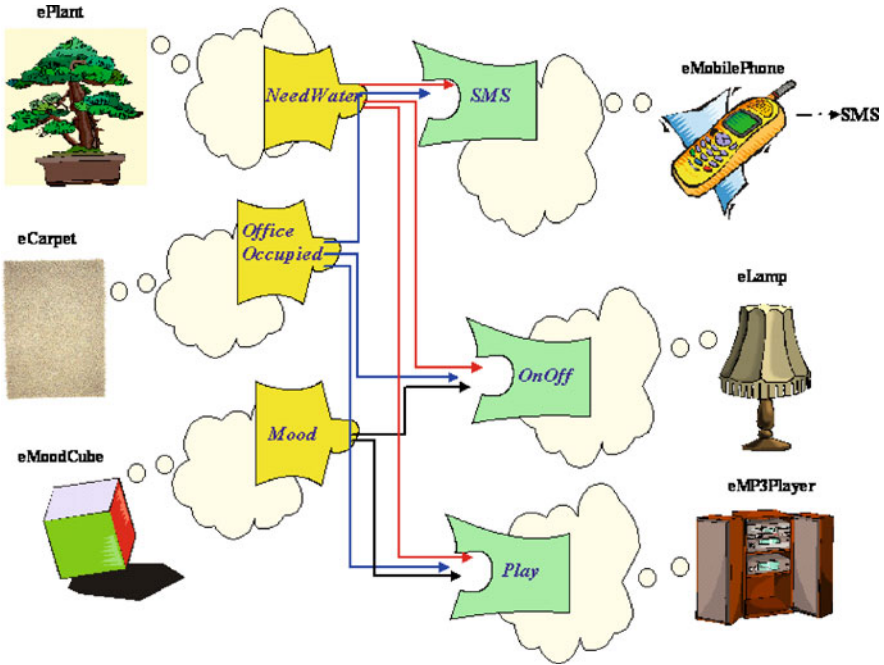


Fig. 20.5 Plugs and synapses for the implementation of the smart plant application

We note that for the *ePlant*, *eCarpet* and *eMoodCube*, the evaluation of the state, service or capability is based on a *local decision making scheme*, because the assessment logic depends only on knowledge that is managed by the specific component through its attached sensor network and rule-base. On the other hand, the service provided by the *eMobilePhone*, *eLamp* and *eMP3Player* depends on a *global decision making scheme*, because the rules that govern the decision to offer a service have to take into account the state and capability information of several eEntities. For example, to decide whether to make the *eLamp* blink (as a visual notification service to the user), we have to take into account the state of the *ePlant*, provided by the *NeedWater* plug, the capability of the *eCarpet* to sense the presence of the user in the office, provided by the *OfficeOccupied* plug, and the capability of the *eMoodCube* to map the mood of the user, through the *Mood* plug. Thus, to turn on or off the *eLamp* we have to define a rule that takes into account all the above plugs.

The following table summarizes the properties, plugs and operation rules (functional schemas) of each eEntity participating in the Smart Plant application. This knowledge is part of the application ontology defined for the application. We have omitted information, such as physical properties, or other plugs, which may reflect services that are not required by the specific application (Table 20.1).

By specifying the rule structure and semantics in an ontology that defines various parameter and state types, as well as the arguments that the rules are based upon,

Table 20.1 Smart plant application ontology configuration (part of)

eEntity	Properties	Plugs	Functional schemas
ePlant	Determining the state of the plant, whether the plant needs irrigation or not.	NeedWater: {OUT Boolean}	<pre> PlantTemp ← read(Thermistors) SoilHumidity ← read(MoistureProbe) AmbientTemp ← read(Thermistors) IF PlantTemp - AmbientTemp > ePlant.TempThreshold OR SoilHumidity < ePlant.HumidityThreshold THEN NeedWater ← TRUE ELSE NeedWater ← FALSE SensorArray ← read(SensorNetwork) OfficeOccupied ← FindMovementDirection(SensorArray) </pre>
eCarpet	The carpet is placed at the entrance of the office. As the user enters or leaves he/she is forced to step over the carpet. The eCarpet monitors the direction of this movement and updates its plug accordingly.	OfficeOccupied: {OUT Boolean}	
eMoodCube	As the moodCube is turned it changes its color and represents user's mood. Possible selections are: <ul style="list-style-type: none"> • DO_NOT_DISTURB, • NOTIFY_VISUAL, • NOTIFY_ACOUSTIC 	Mood: {OUT Enumeration}	<pre> Position ← read(Sensors) Mood ← MapPositiontoMood(Position) </pre>
eMobilePhone	Send SMS Service (S1)	SMS: {IN Boolean}	<pre> IF ePlant.NeedWater AND NOT eCarpet.OfficeOccupied THEN S1() IF ePlant.NeedWater AND eCarpet.OfficeOccupied AND eMoodCube.Mood = NOTIFY_VISUAL THEN S2(BLINK) IF ePlant.NeedWater AND eCarpet.OfficeOccupied AND eMoodCube.Mood = NOTIFY_ACOUSTIC THEN S3(A user-defined audio message) </pre>
eLamp	Light service (S2)	OnOff: {IN Enumeration}	
eMP3Player	Playing message service (S3)	Play: {IN Boolean}	

we can use the ontology to verify the validity of the rules. This also facilitates the inclusion of context parameters in rules, since we know the rule structure and the value types of different arguments. Furthermore, the use of ontological descriptions allows heterogeneous entities to interoperate and interact with one another in a way dictated by the application domain under examination.

20.5.4 Semantic-Based Service Discovery

Service and resource discovery will play an integral role in the realization of Ubiquitous computing systems. Since artifacts are resource limited, they must be able to discover and use the services of neighboring devices. A service discovery mechanism is needed so that if a synapse is broken, e.g. because of an artifact failure, another artifact that offers a semantically equivalent service could be found. In the example application scenario discussed previously, suppose that the synapse between *ePlant* and *eLamp* is broken because of *eLamp* failure. Then, a new artifact having a property that provides the service “light” must be found. Therefore, for a Ubiquitous computing environment the infrastructure must be enhanced to provide a semantic-based service discovery, so that it is possible to discover all the relevant services.

Since for the Ubiquitous computing applications a semantic service discovery mechanism is useful and the replacement of artifacts depends on the services that artifacts offer, a service classification is necessary. In order to define such a service classification we first identified some services that various artifacts may offer; for the application scenario discussed indicative services are presented in

Table 20.2. From these it is clear that the services offered by artifacts depend on artifacts physical characteristics and/or capabilities and their attached sensors and actuators.

Next we had to decide how we should classify the services. The classification proposals that we elaborated are the following: by object category, by human senses and based on the signals that artifacts sensors/actuators can perceive/transmit. We decided to combine these proposals so that to describe a more complete classification. So we initially defined the following elementary forms of signals that are used: sonic, optic, thermal, electromagnetic, gravity and kinetic. These concepts are

Table 20.2 Services that may be offered by artifacts

Artifact	Offered services
ePlant	needs water yes/no, needs nutrients yes/no, species, other physical characteristics.
eCarpet	object on it yes/no, objects' position, direction, pressure, weight, frequency
eMoodCube	current position
eMobilePhone	send sms, send email, make phone call, get calendar, get contacts
eLamp	switch on/off, light, heat
eMP3Player	sound, sound volume, kind of music, play/pause/stop, next/previous track

divided into lower level services (subclasses); e.g. the sonic service may be music, speech, environmental sound, and noise. Additionally services may have a set of properties; e.g. sonic can have as properties the volume, the balance, the duration, the tone, etc. Finally we enriched this classification by adding services relevant to environmental information, like humidity and temperature.

We have defined a lightweight Resource Discovery Protocol for eEntities (eRDP) where the term resource is used as a generalization of the term service. The protocol makes use of typed messages codified in XML. Each message contains a header part that corresponds to common control information including local IP address, message sequence number, message acknowledgement number, destination IP address(es) and message type identification. The prototype was written in Java using J2ME CLDC platform. kXML is used for parsing XML messages.

One of the ontology goals is to describe the services that the artifacts provide and assist the service discovery mechanism. In order to support this functionality, the Ontology Manager provides methods that query the application ontology for the services that a plug provides as well as for the plugs that provide a specific service. Therefore, the Ontology Manager provides to the calling process the necessary knowledge (which is retrieved from the artifact) that is relevant to the artifact services, in order to support the service discovery mechanism. Similarly the Ontology Manager can answer queries for plug compatibility and artifact replace-ability.

Let's return to the scenario discussed above, where the synapse between *ePlant* and *eLamp* is broken. We said that when this happens, the system will attempt to find a new artifact having a plug that provides the service "light". The system software is responsible to initiate this process by sending a message for service discovery to the other artifacts that participate in the same application or are present in the surrounding environment. This type of message is predefined and contains the type of the requested service and the service's attributes. A description of the *eLamp* service is shown in Fig. 20.6.

When the system software of an artifact receives a service discovery message, it forwards the message to the Ontology Manager. Let's assume that the artifact *eBunny* is available and that this is the first artifact that gets the message for service discovery. The *eBunny* Ontology Manager firstly queries the application ontology of *eBunny* in order to find if this artifact has a plug that provides the service "light". If

```

<res_spec>
  <res name> eLamp </ res name>
  <res classification> light </res classification >
  <res id> eRDP:PLUG:CTI-RU3-eLamp-ONOFF_PLUG </res id >
  <res location> 150.140.30.5 </res location>
  <res data> <attrName="power" type="bool" value="false"
  <attrName="luminocity" type="integer", value="10"
  </res data>
  <res timestamp> 4758693030 </res timestamp>
  <res expiry> Never </res expiry>
</res_spec>

```

Fig. 20.6 XML description of eLamp service

we assume that the *eBunny* has the plug “*LampBlink*” that provides the service light, the Ontology Manager will send to the system software a message with the description of this service. If such a service is not provided by the *eBunny*, the Ontology Manager queries the *eBunny* application ontology in order to find if another artifact, with which the *eBunny* has previously collaborated, provides such a service. In case of a positive answer it returns as a reply the description of this service. If the queried artifact, in our example the *eBunny*, has no information about an artifact that provides the requested service, the control is sent back to system software, which is responsible to send the query message for the service discovery to another artifact.

20.6 Conclusions

The ontology and the context management framework that we developed sufficiently supports the composition of context-aware Ubiquitous computing applications from everyday enhanced physical objects and it also address a number of key issues of such applications like application model dynamic adaptability and semantic service discovery. The context model that we selected for both these Ubiquitous computing applications is the same ontology-driven model.

Future Ubiquitous computing environments will involve hundreds of interacting and cooperating devices ranging from unsophisticated sensors to multi-form actuators. Although the majority of these devices may have limited resources (computation, memory, energy, etc) or may be only oriented to certain tasks, their collective behavior that results from local interactions with their environment may cause coherent functional global patterns to emerge. Hence, the combination and cooperation of locally interacting artifacts with computing and effecting capabilities may trigger the continuous formation of new artifact ecologies that provide services not existing initially in the individuals and exhibit them in a consistent and fault-tolerant way. As these societies are dynamically reconfigured aiming at the accomplishment of new or previous related tasks, their formation heavily depends not only on space and time but also on their context of previous local interactions, previous configured teams, successfully achieved goals or possibly failures. This means that in order to initially create, manage, communicate with, and reason about, such kinds of emergent ecologies, we need somehow to model and embed to these entities social memory, enhanced context memory, and shared experiences. One step to this end is the design and implementation of evolving multidimensional ontologies that will include both nonfunctional descriptions, and rules and constraints of application, as well as aspects of dynamic behavior and interactions.

References

- Bergman, E. 2000. *Information appliances and beyond*. San Francisco, CA: Morgan Kaufmann Publishers.
- Bunge, M. 1977. *Treatise on basic philosophy: Volume 3: Ontology I: The furniture of the world*. Dordrecht: Reidel.

- Bunge, M. 1979. *Treatise on basic philosophy: Volume 3: Ontology II: A world of systems*. Dordrecht: Reidel.
- Chen, H., T. Finin, and A. Joshi. 2003. An ontology for context aware pervasive computing environments. *Knowledge Engineering Review* 18(3):197–207.
- Christopoulou, E., and A. Kameas. 2005. GAS ontology: An ontology for collaboration among ubiquitous computing devices. *International Journal of Human-Computer Studies* 62(5):664–685.
- De Bruijn, J. 2003. Using ontologies – enabling knowledge sharing and reuse on the semantic web. Technical Report DERI-2003-10-29. Austria: Digital Enterprise Research Institute (DERI).
- De Paoli, F., and M. Loregian. 2006. Context-aware applications with distributed ontologies. In Proceedings of the CAISE*06 Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS '06), CEUR Workshop Proceedings 242, eds. M.C. Norrie, S. Dustdar, and H. Gall, 869–883.
- Dey, A.K. 2001. Understanding and using context, personal and ubiquitous computing. *Special Issue on Situated Interaction and Ubiquitous Computing* 5(1):4–7.
- Dey, A.K., D. Salber, and G.D. Abowd. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal* 16(2–4):97–166.
- Dearle, A., G.N.C. Kirby, R. Morrison, A. McCarthy, K. Mullen, Y. Yang, R.C.H. Connor, P. Welen, and A. Wilson. 2003. Architectural support for global smart spaces. In Proceedings of the 4th International Conference on Mobile Data Management (MDM'03), 153–164. London: Springer.
- Disappearing Computer Initiative. 2007. <http://www.disappearing-computer.net/>. Accessed on 2 Aug 2007.
- Dobson, S., and P. Nixon. 2004. More principled design of pervasive computing systems. In Proceedings of Engineering for Human-Computer Interaction and Design, 292–305. Berlin: Springer.
- Drossos, N., C. Goumopoulos, and A. Kameas. 2007. A conceptual model and the supporting middleware for composing ubiquitous computing applications. *Journal of Ubiquitous Computing and Intelligence American Scientific Publishers (ASP)* 1(2):1–13.
- Ejigu, D., M. Scuturici, and L. Brunie. 2007. CoCA: A collaborative context-aware service platform for pervasive computing. In Proceedings of the International Conference on Information Technology, IEEE computer society, 297–302.
- Goumopoulos, C., E. Christopoulou, N. Drossos, and A. Kameas. 2004. The PLANTS system: Enabling mixed societies of communicating plants and artefacts. In Proceedings of the 2nd European symposium on Ambient intelligence (EUSAI 2004), eds. P. Markopoulos, B. Eggen, E.H.L. Aarts, and J.L. Crowley, 184–195. London: Springer.
- Gruber, R. 1993. A translation approach to portable ontology specification. *Knowledge Acquisition* 5(2):199–220.
- Guizzardi, G., H. Herre, and G. Wagner. 2002. On the general ontological foundations of conceptual modeling. In Proceedings of the 21st International Conference on Conceptual Modeling, eds. S. Spaccapietra, S.T. March, and Y. Kambayashi, 65–78. London: Springer.
- Henricksen, K., J. Indulska, and A. Rakotonirainy. 2002. Modeling context information in pervasive computing systems. In Proceedings of the 1st International Conference on Pervasive Computing (Pervasive 2002), eds. F. Mattern, and M. Naghshineh, 167–180. London: Springer.
- Jess, 2007. Rule engine for the java platform. <http://herzberg.ca.sandia.gov/jess/>. Accessed on 2 Aug 2007.
- Kameas, A., S. Bellis, I. Mavrommati, K. Delaney, M. Colley, and A. Pounds-Cornish. 2003. An architecture that treats everyday objects as communicating tangible components. In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom03), 115–122. IEEE CS Press.
- Kagal, L., V. Korolev, H. Chen, A. Joshi, and T. Finin. 2001. Centaurus: A framework for intelligent services in a mobile environment. In Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCSW'01), 195–201. IEEE Computer Society.

- Kindberg, T., J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. 2000. People, places, things: Web presence for the real world. Technical Report HPL-2000-16, HP Labs.
- Milton, S.K., and E. Kazmierczak. 2004. An ontology of data modelling languages: A study using a common-sense realistic ontology. *Journal of Database Management* 15(2):19–38.
- Norman, D. 1999. *The invisible computer*. Cambridge, MA: MIT Press.
- Ranganathan, A., and R. Campbell. 2003. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing* 7(6):353–364.
- Rosemann M., and P. Green. 2000. Integrating multi-perspective views into ontological analysis. In Proceedings of the 21st International Conference on Information Systems, 618–627. Brisbane: Association for Information Systems.
- Rosemann M., and P. Green, and M. Indulska. 2004. A reference methodology for conducting ontological analyses. In Proceedings of Conceptual Modeling – ER 2004, 110–121. London: Springer.
- Russell, S., and P. Norvig. 2003. *Artificial intelligence: A modern approach*, 2nd edition. Upper Saddle River, NJ: Prentice Hall.
- Gu, T., X. H. Wang, H. K. Pung, and D.Q. Zhang. 2004. An ontology-based context model in intelligent environments. In Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04), 270–275. Society for Computer Simulation.
- Strang, T., and L. Linnhoff-Popien. 2004. A context modelling survey. Proceedings of the 1st International Workshop on Advanced Context Modelling, Reasoning and Management. <http://www.itee.uq.edu.au/~pace/cw2004/Paper15.pdf>.
- Strang, T., L. Linnhoff-Popien, and K. Frank. 2003. CoOL: A context ontology language to enable contextual interoperability. In Proceedings of the 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003), 236–247. Springer.
- Wand, Y., and R. Weber. 1990. An ontological model of an information system. *IEEE Transactions on Software Engineering* 16(11):1282–1292.
- Wang, X.H., D.Q. Zhang, T. Gu, and H.K. Pung. 2004. Ontology based context modeling and reasoning using OWL. In Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops. 18. IEEE Computer Society.
- Weber, R. 1997. Ontological foundations of information systems. Coopers & Lybrand Accounting Research Methodology. Monograph No. 4.
- Weiser, M. 1991. The computer for the 21st century. *Scientific American* 265(3):94–104.

Chapter 21

Category Theory as a Mathematics for Formalizing Ontologies

Michael John Healy

21.1 Introduction

This chapter has a twofold purpose. First, since a knowledge of category theory is uncommon, it serves as a tutorial in this subject. The intent is to provide a sufficient background for the chapters by Baianu and Poli, Johnson and Rosebrugh, Kalfoglou and Schorlemmer, Kent, and Vickers, and in this spirit examples are provided along with explanation. Second, since a knowledge of both the potential and current application of category theory is also uncommon, this Introduction attempts to motivate the tutorial by providing some discussion of why this subject is worth the concentrated effort required to learn about it.

Category theory is a recent branch of mathematics and, as one might expect, expresses a special meaning of the term “category” that differs somewhat from the common informal use. The common use derives from that of Aristotle: A class or collection of things all of one type, where typicality is determined by a mental representation or concept. To Aristotle a category had an overall, abstract sense of that which determines membership within it: substance, quality, position, etc. would be categories determined by explanation and argument among philosophers. In much modern work, a concept is often represented as a set of “features” such as shape, color, or function. In the classical view, all examples of a concept – often called category members – are equally good representatives of it. However, recent research has led to the view that members have degrees of typicality (Medin, 1989); for example, a domestic cat may be seen as a more typical feline than a tiger. In fact, the structure within a category determined by relationships among members is a subject of much current research. Ultimately, this leads back to the more classical idea that concepts are closed systems of knowledge, where items are categorized based upon explanation rather than typicality alone. In the modern view, the explanatory determinant or concept for a category determines an internal structure given by relationships

M.J. Healy (✉)

Department of Electrical and Computer Engineering, University of New Mexico,
Seattle, WA 98125, USA
e-mail: mjhealy@ece.unm.edu

among its members, which may in turn suggest degrees of typicality. Put another way, the relationships reflect the fundamental knowledge expressed in the concept that explains why something is a member of the category; it is included because of its sharing of certain features with certain other somethings.

Since ontology is often regarded as a categorization of that which exists, issues involving structure in categories are important. Since more than one category may be involved, relationships between categories are also worth considering, and thus there is a structure determined by the categories and their relationships as well. This implies a two-level notion of structure: between and within categories. But why stop there? In modern research, there are hierarchies of categories within categories, so there can be many levels of structure. In fact, as we shall see, relationships can themselves have relationships, which leads to the notion that there are categories whose members are relationships.

Further, there can be levels within a structure; that is, the relationships within many categories form a hierarchical structure. Often, this takes the form of a hierarchy of abstractions or – in the opposite direction – specializations. To give a taxonomic example of this, consider a system of categories of types of animals, where the categories are related by superordination; for example, the category of birds is a superordinate of the category of robins. Categories of this kind, related by superordination, form a hierarchical structure which is also a category. The superordinate, bird category is more abstract in the sense that fewer descriptive features are needed to characterize birds than would be required for a full characterization of robins. Therefore, the superordinate category, because its concept imposes fewer constraints on membership, has relatively more members. The perceptual and other features defining typicality of members of a “basic” category such as “dog” or “domestic cat” are greater in number than those determining membership in a superordinate such as “mammal”, where features are equivalenced or deleted or specific descriptions such as “has retractable claws” are replaced by an abstraction such as “has nails”.

Another important facet of categories is the notion of relationships between structures. Consider the superordination relationship between the categories “domestic cats” and “mammals”. Within this relationship, each member of the category “domestic cat” corresponds to (has a manifestation as) a member of the category “felines”, and each feline (there are many more of these, including tigers, etc.) has in turn a manifestation as a “mammal”; by ignoring the constraints specializing “domestic cat” category members, we obtain members of the category “felines”, and further ignoring the constraints specifying felines, one obtains members of the category “mammals”. The fact that we can associate each member of a category with a unique member of a superordinate category implies that the superordination relation on a pair of taxonomic categories is an example of what mathematicians call a *mapping*. Notice that in the example, the superordination relation is transitive: Each domestic cat corresponds to a feline and each feline corresponds to a mammal, and, hence, each domestic cat corresponds to a mammal. We say that the individual superordination relationships are *composable*. As we shall see, this is a special case of a fundamental property of structural relationships.

As it turns out, all that has been said in the foregoing can be formalized – or approximated by a formalization, if you prefer – in the mathematical discipline of category theory. Indeed, category theory is the mathematical theory of structure, and its greatest significance lies in its capacity to express relationships between structures. An elementary example of this is the superordination relation between categories discussed in the preceding paragraph. As was mentioned initially, however, the mathematical notion of a category is different from that in common use, and would be if for no other reason than that it must be precise according to the usual criterion of mathematical rigor. It differs in other ways as well. The use of the term “category theory” arose out of studies of the relationship between different kinds of mathematical structure. Specifically, in algebra, one studies operations on numbers – or for permutation groups on geometric shapes or on particles in physics, one studies symmetries or orientations of a shape or the allowed configurations (manifestations, states) of a particle in a multi-dimensional configuration space. In topology, on the other hand, one studies continuity, as in the continuous functions of calculus. There are different algebraic and topological systems, and operations and continuity are two wholly different concepts – and are two sides of mathematics. Yet, certain algebras have structures – multiples of an integer, for example – that are directly related to certain topological structures – a “rubber-sheet” space such as a bagel or a tire, which includes a hole, for example. The study of the relationship between these two kinds of structure was the beginning of category theory. Each kind of structure – algebraic and topological – forms a mathematical category. Each category has members, called objects – algebras in one, topological spaces in the other. These are related in pairs – an algebra to a space. But they are categories because of their internal structure. Algebras are related; for example, an Abelian group, exemplified by the operation of addition on the integers, is also a group, which type includes also permutation groups which generally behave nothing like the integers other than sharing the basic operation of a group (addition in the integer example, concatenation of permutations in the other). Spaces are also related, by continuous functions. Because of these facts, the importance of category theory stems from the importance it attaches to the following concept: A relationship between two categories involves more than just a pairing of objects from each; it is intimately involved with the pairing of the relationships between objects in the two categories. This will be explained in greater detail, with examples drawn from computer science. For now, realize that there is a distinction in the use of the term “category” in this and the other mathematical chapters. At the most general level, category theory is useful because it provides a vehicle for the formalization of ontologies with mathematical rigor. Formalisms based in mathematical logic allow individuals to be represented with variables and constants in formulas. These are accompanied by a model-theoretic foundation that allows an analysis of the instances that satisfy the closed formulas, or sentences, of a theory in the logic. The sentences separate the instances, called models, into categories and state constraints on them. This provides a structure on and within the models – classes of individuals, functions mapping individuals between classes, and sub-classes defined by predicates.

Kalfoglou and Schorlemmer (Schorlemmer and Kalfoglou, 2005) view an ontology as a theory in formal logic. They introduce a category-theoretic notion of semantic alignment between ontologies, by which they mean a system of relationships (morphisms – see the next section) which they call an information system. An information system shows how the terms in different ontologies are associated. Category theory has proven not only a powerful medium for expressing structures and their relationships (theories, for example), but also has led to a new paradigm in their study known as categorical logic. Vickers discusses a formalism in a categorical logic and its accompanying model theory based upon set-theoretic interpretations. This logic has been used for formal specifications for software and for analyzing database semantics (Vickers, 1992). Kent describes the categorical formalism used in the Information Flow Framework (IFF) (Kent, 2003). The latter provides a framework for ontology development and use within category theory. Here, the categories include categories of theories, of models, and items related to formal concept analysis (FCA), making it possible to communicate between the categorical and FCA formalisms. The result is a mathematical system for knowledge representation that can be applied to investigations in ontology. Johnson and Rosebrugh describe an approach for ontology formalization that has proven particularly effective in solving problems for enterprise information systems (Colomb et al., 2001); for example, it has provided a new and more general theory-based and yet practical solution method for the view updating problem in database management (Johnson and Rosebrugh, 2001). Baianu and Poli propose a new paradigm for the study of the many levels of reality and complex systems, from physical (atoms, molecules, etc) to biological to social and beyond. They propose a combination of category theory and novel mathematical frameworks which they call non-Abelian mathematics (Baianu et al., 2007).

There are many applications of category theory, most of them recent. Application areas in addition to those already mentioned include other kinds of logic (Lawvere, 1963; Goguen and Burstall, 1984; Meseguer, 1989; Goguen and Burstall, 1992; Crole, 1993), system theory (Goguen, 1973), software synthesis (Burstall and Goguen, 1980; Jullig and Srinivas, 1993; Williamson and Healy, 2000), the mathematical study of biological systems (Rosen, 1958; Baianu, 1988; Ehresmann and Vanbremeersch, 1997; Gust and Kühnberger, 2005; Healy and Caudell, 2006), and other efforts in the formalization of ontologies (Uschold et al., 1998; Dampney et al., 2001). The following sections provide an initial exploration of category theory.

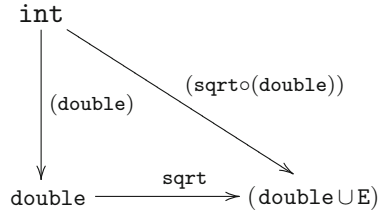
21.2 Categories

Category theory grew out of the mid-1940s investigations of Samuel Eilenberg and Saunders Mac Lane into the relationships between certain kinds of mathematical structures. F. William Lawvere's 1963 paper (Lawvere, 1963) opened up the field of categorical logic and model theory and has led to the widespread use of

category theory in mathematical semantics. The reader interested in acquiring depth in category theory or seeing more examples of its applications is encouraged to consult one of the available introductory texts, including (Mac Lane, 1971; Adamek et al., 1990; Pierce, 1991; Walters, 1991; Crole, 1993; Lawvere and Schanuel, 1995).

The Introduction elaborated on the non-mathematical notion of categories in terms of the notion of structure. Thinking of structure brings a graphical picture to mind; after all, structure is, at the most basic level, a system of interrelated items, where the relationships between items appear as links. Consider, then, the structure consisting of all the subsets of a set. Here, the links express the subset relation, where $X \subseteq Y$ is shorthand for “ X is a subset of or is the same set as Y ”. This means that any element of X is also an element of Y , that is, $x \in X$ implies $x \in Y$. Here is where the graphical notion, with the subset relationships as links, ends. Notice that if also $Y \subseteq Z$, then the transitivity of the \subseteq relation can be used to infer the additional relationship $X \subseteq Z$. The hyponymy relation on noun synsets in WordNet (see Chapter 10, by Fellbaum, this volume) provides another example of transitivity. Transitivity makes a relational system more than a graph; it makes a relation *compositional*. Functions defined on sets also express composition. A function f from X to Y associates each element x of a set X , denoted $x \in X$, with a unique element $y \in Y$ (notice that there is no requirement that a function must map some x to every y , and many x 's may map to one y). This relationship is represented externally to X and Y as $f: X \rightarrow Y$ and internally, or elementwise, by the *function evaluation* $y = f(x)$. A numerical example is given by the expression $y = x^2$, defining a function $f: \mathbf{R} \rightarrow (\mathbf{R}^+ \cup \{0\})$, where \mathbf{R} and $\mathbf{R}^+ \cup \{0\}$ denote the sets of real numbers and nonnegative real numbers, respectively. If X , Y and Z are sets and $f: X \rightarrow Y$ and $g: Y \rightarrow Z$ are functions and $x \in X$, the composition $g \circ f: X \rightarrow Z$ maps x as $(g \circ f)(x) = g(f(x))$, obtaining $g(f(x)) \in Z$ by way of $f(x) \in Y$.

The uses of function composition in mathematics are legion, and so are its uses in computational theory. A simple example is that of data type coercion in computer programming. It is sometimes desirable to use a data item of one type as an argument to a computerized function whose arguments are of a different type. A popular C language programming book (Kernighan and Ritchie, 1988) provides the example of converting a member of the integer data type `int` to a double precision value, of type `double`, when it is desired to find the square root of an integer, where the `sqrt` function in C requires an argument of type `double`. Letting `int` and `double` serve as mathematical names for the sets of computer-representable members of the types `int` and `double`, respectively, let $(\text{double}) : \text{int} \rightarrow \text{double}$ denote mathematically the type conversion function and $\text{sqrt} : \text{double} \rightarrow (\text{double} \cup E)$ denote the square root function. Here, E is a set of elements of an error type, to provide for occurrences such as the domain error in which the `double` argument to `sqrt` is negative. Mathematically, the type coercion followed by `sqrt` evaluation is the composition $(\text{sqrt} \circ (\text{double})) : \text{int} \rightarrow (\text{double} \cup E)$. In category theory, a composition like this is illustrated with a diagram that is said to *commute*:



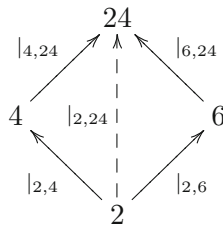
A non-compositional relation can usually be replaced with little or no loss of expressive power by a compositional one. For example, the “is the son (daughter) of” relation can be replaced by the “is a descendant of” or “is a male (female) descendant of” relation. However, the single primitive upon which classical set theory is based – the element relation \in , where for example $x \in X$ – is inherently not compositional in nature. Category theory, on the other hand, is based upon a primitive \longrightarrow that represents a compositional relation, $a \longrightarrow b$ on entities a and b . As a consequence, powerful as it is, set theory has had to share with its newer cousin category theory the role of fundamental mathematical theory.

A category is a mathematical structure consisting of entities of some kind together with the relationships between them that express the structure. Formally, a category consists of *objects* (the entities) and *morphisms* (the relationships), together with a law of composition for the morphisms. Generically, the notation for a morphism of a category C is $f: a \longrightarrow b$, where a and b are objects of C ; a and b are called the domain and codomain, respectively, of f (f and $f: a \longrightarrow b$ are used interchangeably when the context is clear). Given morphisms $f: a \longrightarrow b$ and $g: b \longrightarrow c$ in C , where the domain of g is the codomain b of f , there must exist in C a morphism $g \circ f: a \longrightarrow c$, their *composition*. Composition is associative, meaning that in triples of morphisms which have a domain-to-codomain match by pairs, $f: a \longrightarrow b$, $g: b \longrightarrow c$ and $h: c \longrightarrow d$, the result of successively forming compositions is order-independent, so that $h \circ (g \circ f) = (h \circ g) \circ f: a \longrightarrow d$. Also, for each object a , there is an *identity morphism* $\text{id}_a: a \longrightarrow a$ such that the equations $\text{id}_a \circ g = g$ and $f \circ \text{id}_a = f$ hold for any arrows $f: a \longrightarrow b$ and $g: c \longrightarrow a$.

That this notation is identical with the function composition notation \circ of the preceding section is no coincidence, for in the first categories studied the morphisms were functions with special properties. Categories are still a novelty to many people, while sets are more familiar and, indeed, category theory has a place for them: The category **Set** (or **Sets**, to some authors) has sets as its objects and (total) functions as its morphisms. In the C programming example of the previous section, the sets `int`, `double` and `(double ∪ E)` are objects of **Set** and the functions `(double)`, `sqrt`, and `(sqrt ∘ (double))` are morphisms. Another example of a category is given by any lattice: There is exactly one morphism for each pair of objects a, b for which $a \leq b$ holds. The transitive property: $a \leq b$ and $b \leq c$ implies $a \leq c$, yields the composition operation, which is easily seen to be associative, and the identity

morphism for a is $a \leq a$. Lattices also provide an elementary example of two mathematical constructs built on composition, colimits and limits, which in a lattice are the join or least upper bound and meet or greatest lower bound. This is because a lattice is a partial order in which every pair of elements (a, b) has a join $a \vee b$ and also a meet $a \wedge b$. These two examples, any lattice and **Set**, represent two extremes in the following sense. A lattice has at most one morphism in a single direction between two distinct elements. For two objects of **Set**, however, there can be many morphisms in *both* directions.

The equations of category theory are *commutative diagrams* (or *commuting diagrams*). Informally, a diagram in C is simply a collection of objects and morphisms of C . The domain and codomain objects of a morphism are considered part of it, so these are automatically included in the diagram if the morphism is included. In a commutative diagram, any two morphisms with the same domain and codomain, where at least one of the morphisms is the composition of two or more diagram morphisms, are equal. This is a formalization of the notion that there can be more than one way to obtain a particular relationship by passing through intermediaries. For a simple example, consider the category $\mathbb{N}_+^|$ in which the objects are nonzero natural numbers and in which there is a morphism $|_{n,m}: n \rightarrow m$ exactly when n is a divisor of m , $n | m$. In fact, this is a lattice in which composition is just the transitive property of the divisor relation and the join and meet of two elements are their least common multiple (lcm) and greatest common divisor (gcd), respectively. The diagram commutes, expressing the fact that the knowledge that $2 | 24$ can be obtained



either by knowing that $2 | 4$ and $4 | 24$, or that $2 | 6$ and $6 | 24$. The diagram expresses this as an equality of compositions, $|_{4,24} \circ |_{2,4} = |_{2,24} = |_{6,24} \circ |_{2,6}$. Here, the common domain of the two compositions is 2 and their common codomain is 24, and they are equal because there is only one divisor relationship between 2 and 24. Of course, the lcm of 4 and 6 is 12, or $4 \vee 6 = 12$, which would yield a different commutative diagram. In general, a commutative diagram is a graph-like expression of a system of equations involving compositions of morphisms, a useful way of visualizing a system of constraints on mathematical structures in a category.

The principle of duality is a fundamental notion in category theory. The dual or opposite C^{op} of a category C has the same objects but the arrows and compositions are reversed, $(g \circ f)^{op} = f^{op} \circ g^{op}$. The *dual of a statement* in category theory is the statement with the words “domain” and “codomain”, “initial” and “final”, and the compositions reversed. If a statement is true of a category C , then its dual is

true of C^{op} . If a statement is true of all categories, the dual statement is also true of all categories because every category is dual to its dual. Roughly speaking, “half the theorems of category theory are obtained for free”, since proving a theorem immediately yields its dual as an additional theorem; see any of (Adamek et al., 1990; Pierce, 1991; Mac Lane, 1971).

In addition to the widely-used notion of duality, category theory provides a mathematically rigorous notion of “isomorphism”, a term which is often used in a loose, intuitive sense. One sometimes hears a statement such as “the two [concepts, data types, program constructs, etc.] are in some sense isomorphic”. If the entities under discussion can be formalized as objects in a category, one can make such statements with mathematical rigor. If a, b are objects of a category C such that there exist arrows $f: a \rightarrow b$ and $g: b \rightarrow a$ with $f \circ g = \text{id}_b$ and $g \circ f = \text{id}_a$, then the morphism f is called an *isomorphism* (as is g also) and g is called its *inverse* (and f is called the inverse of g), and the two objects are said to be isomorphic. The property of an identity morphism ensures that isomorphic objects in a category are interchangeable in the sense that they have the same relationships with all objects of the category. In the category **Set** an isomorphism is a bijective function or bijection $f: X \rightarrow Y$. That is, $f(x) = f(x')$ implies $x = x'$ (f is one-to-one, or injective) and for all $y \in Y$, there exists $x \in X$ with $y = f(x)$ (f is onto, or surjective). In \mathbb{N}_1^+ , the only isomorphisms are identities, $n \mid n$.

An *initial object* of a category C is an object i that serves as the domain of a unique morphism $f: i \rightarrow a$ for every object a of C . A terminal object t is the dual notion, obtained by reversing arrows in the definition of i – that is, it serves as the codomain of a unique morphism $f: a \rightarrow t$ for every object a of C . It is easy to show that all initial objects in a category are isomorphic, and ditto for terminal objects. For initial objects, suppose that i, i' are initial in C . Then, applying initiality to each object, there must be unique morphisms $f: i \rightarrow i'$ and $f': i' \rightarrow i$. But the compositions $f' \circ f: i \rightarrow i$ and $f \circ f': i' \rightarrow i'$ must be unique as well, implying that $f' \circ f = \text{id}_i: i \rightarrow i$ and $f \circ f' = \text{id}_{i'}: i' \rightarrow i'$. Therefore, i and i' are isomorphic. Terminal objects in C are isomorphic by duality. The empty set, \emptyset , is the single initial object of **Set**, since for any set a there is a unique function $f: \emptyset \rightarrow a$ whose domain is \emptyset and whose codomain is a , namely, the vacuous function, since there are no elements in \emptyset to map to an element of a . There is an infinite number of terminal objects in **Set**, namely the singletons $\{x\}$, since there is a single function $f: a \rightarrow \{x\}$ mapping the elements of any set a to x .

21.3 Limits, Colimits, and Concepts as Theories

An important use of commutative diagrams and terminal and initial objects is in the definition of a *limit* of a diagram and the dual type of quantity, a *colimit*. In the many important categories in which they exist universally, colimits express complex structures in terms of simpler sub-structures. Colimits will be described here in some detail because of the author’s familiarity with them in applications; limits are equally

important where they exist, and they can be described by dualizing the following discussion. See Mac Lane (1971) for an in-depth mathematical treatment of limits and colimits.

Consider an example Δ of a diagram in a category C as shown in Figs. 21.1 and 21.2, with objects a_1, a_2, a_3, a_4, a_5 and morphisms $f_1 : a_1 \rightarrow a_3, f_2 : a_1 \rightarrow a_4, f_3 : a_2 \rightarrow a_4, f_4 : a_2 \rightarrow a_5$. The cone-like extensions of Δ shown, such as K in Fig. 21.1, have an *apical object* b and *leg morphisms* $g_i : a_i \rightarrow b$ ($i = 1, \dots, 5$), where for each object of Δ there is a g_j having that object as domain and b as codomain. These have the property that the triangles formed by each f_i and two of the g_j commute, $g_1 \circ f_1 = g_2 = g_3 \circ f_2$ and $g_3 \circ f_3 = g_4 = g_5 \circ f_4$. The structure K is called a *cocone* for Δ . In general, a diagram can have many cocones or it can have few or none, depending upon the available objects and morphisms in C . Given cocones K' and K'' for Δ in Fig. 21.2, with respective apical objects b', b'' and leg morphisms g'_i and g''_i ($i = 1, \dots, 5$), a *cocone*

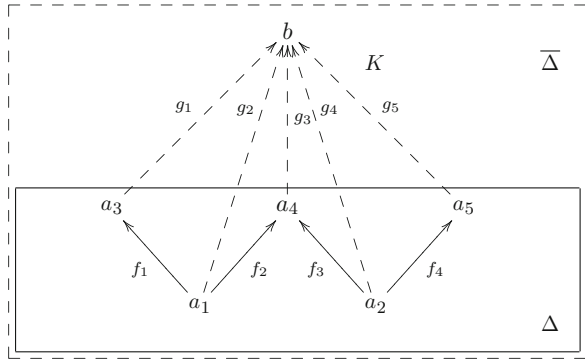


Fig. 21.1 A cocone for a diagram Δ

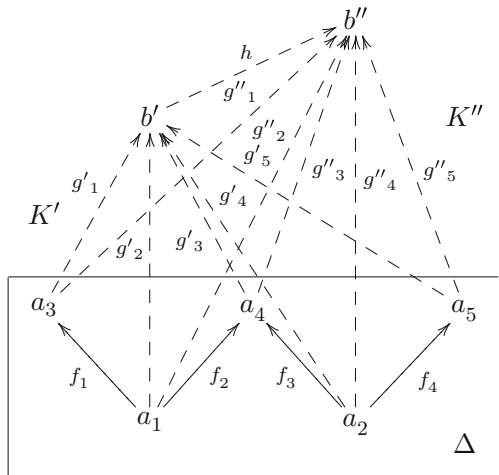


Fig. 21.2 A cocone morphism $h: K' \rightarrow K''$ in coc_{Δ}

morphism with domain K' and codomain K'' is a C -morphism $h: b' \rightarrow b''$ having the property

$$g''_i = h \circ g'_i \quad (i = 1, \dots, 5). \tag{21.1}$$

That is, h is a factor of each leg morphism g''_i of K'' with respect to the corresponding leg morphism g'_i of K' . Re-using the symbol h for notational efficiency, we denote the cocone morphism determined by the C -morphism h as $h: K' \rightarrow K''$.

With morphisms so defined, and the composition for cocone morphisms following directly from the composition for C -morphisms, the cocones for Δ form a category, \mathbf{coc}_Δ . A colimit for the diagram Δ is an initial object K in the category \mathbf{coc}_Δ . That is, for every other cocone K' for Δ , there exists a unique cocone morphism $h: K \rightarrow K'$. If the cocone in Fig. 21.1 is a colimit, the original diagram Δ is called its *base diagram* and $\overline{\Delta}$, formed by adjoining K to Δ , is its *defining diagram*. Note that, as all initial objects are isomorphic, all colimits for a given base diagram are isomorphic.

Limits are the dual notion to colimits; that is, the one notion is obtained from the other by “reversing the arrows” and interchanging “cocone” and “cone” and “initial” and “terminal”. Limits are as important as colimits, but many of their applications differ. A very important kind of limit, called a *product*, occurs in many categories; its base diagram is discrete, having objects but no morphisms. The limit of a diagram with two objects a and b is illustrated in Fig. 21.3. The limit cone has an apical object denoted $a \times b$; its cone leg morphisms, denoted $\pi_a: a \times b \rightarrow a$ and $\pi_b: a \times b \rightarrow b$, are called *projections*. An arbitrary cone with apical object c and leg morphisms $f: c \rightarrow a$ and $g: c \rightarrow b$ is shown to illustrate the terminal cone property of the product. That is, there is a unique morphism denoted $(f, g): c \rightarrow a \times b$ such that $f = \pi_a \circ (f, g)$ and $g = \pi_b \circ (f, g)$. Products are the familiar cartesian products of sets in \mathbf{Set} , which has products for all discrete diagrams including infinite ones.

A theory morphism shows how the symbols of its domain theory map into corresponding quantities in its codomain theory, transforming the axioms of its domain

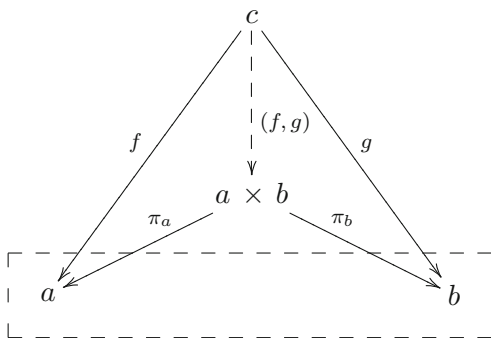


Fig. 21.3 A product is a limit for a discrete diagram

into axioms or theorems of its codomain. Let us view an example which, although almost trivial, illustrates this idea. The following is a theory expressed in a sorted first-order logic:

```
Theory Red
  sorts Colors, Items
  op has_color: Items, Colors -> Boolean
  const red: Colors
  Axiom red-is-expressed is
  exists (it: Items) (has_color (it,red))
end
```

It has sorts for colors and items of an unspecified nature; sorts are an alternative to the use of predicates for distinguishing different classes of objects expressed in a theory. The sorts are to be interpreted as sets in any model of the theory (assuming that the theory is to be interpreted using sets). Operation symbols such as `has_color` are to be interpreted as total functions. The operation `has_color` has two arguments, a color and an item, and maps a pair of these to a Boolean value indicating truth or falsehood (“the item x has the color y ”, true or false). Of course, there is really no color theory here, so the models of this theory can be practically anything. The single axiom decrees that there exists an item that is red, where red is a constant of sort Color. Next, consider the following theory:

```
Theory RYG
  sorts Colors, Apples
  op has_color: Apples, Colors -> Boolean
  const red: Colors
  const yellow: Colors
  const green: Colors
  Axiom some-apple-colors is
  exists (x, y, z: Apples)
    (has_color (x,red))
    and (has_color (y, yellow))
    and (has_color (z, green))
end
```

The theory RYG is just slightly more complex and specialized. Its single axiom decrees that there exists an apple of each one of the three colors red, yellow, green (or not, depending upon whether a model of this theory really has anything to do with apples or colors; in any case, it says that there are at least three items each of which satisfies a two-argument predicate together with a distinct one of three items of a different sort). Now consider a morphism from Red to RYG, expressed as follows:

Morphism s :	Colors	\mapsto	Colors
	Items	\mapsto	Apples
	has_color	\mapsto	has_color
	red	\mapsto	red

This morphism appears to do nothing because each symbol is mapped to itself. This can be very important, however, because it expresses the importation of Red into RYG. In order for s to be a theory morphism, the axiom of Red must map, through symbol translation, to an axiom or a theorem of RYG. Trivially, it maps to a theorem, so s is a theory morphism. But there is more than one morphism with Red as domain and RYG as codomain. Consider the following perfectly valid morphism:

Morphism s' :	Colors	\mapsto	Colors
	Items	\mapsto	Apples
	has_color	\mapsto	has_color
	red	\mapsto	yellow

This extremely simple example can be used to illustrate an important point. A concept hierarchy can be expressed as a lattice, where a lattice link (a morphism) simply means that its codomain is at least as abstract as its domain. However, a theory category is more expressive than a lattice. Its “links” are theory morphisms, oriented in the direction of specialization, opposite abstraction. Each morphism expresses more than just the fact that its codomain is at least as specialized as its domain; it expresses *how, in what sense* it is specialized. Therefore, using a theory category to express a concept hierarchy conveys significantly more information about the hierarchical structure.

The effect of this on analysis can be seen in the categorical constructs made available. First, it allows concepts to be “calculated” as colimits of relatively abstract concepts by “blending” theories along shared sub-theories in a diagram. In Fig. 21.1, for example, let a_3, a_4 and a_5 be theories such as RYG and let a_1, a_2 represent sub-theories such as Red. The colimit theory b is a specialization derived from the diagram. The “blending” is a consequence of the commutative triangles in the defining diagram: where two triangles have a common side, such as $g_2: a_1 \rightarrow b$, the image of the domain a_1 of g_2 must appear only once within the combination of a_3, a_4 within b . This cannot be accomplished simply by forming what some call a union of theories, because such a union entails no sharing, only a disjoint combination. In fact, the union of theories is just the colimit of a discrete diagram, which, being the dual notion to a product, is called a *coproduct*. The colimit provides an example of the idea that certain philosophical notions can be formalized with mathematical rigor, at least in so far as the concept representation of the theory category is a philosophically useful one. In this example, the base diagram morphisms f_1, f_2 and f_3, f_4 can be seen as sharings of ontological commitment to theory a_1 by theories a_3, a_4 and to a_2 by a_4, a_5 . For examples, see Jullig and Srinivas (1993; Healy and Williamson (2000). Colimits have a history of use in system theory, software synthesis and knowledge representation (Goguen, 1973; Burstall and Goguen, 1977; Goguen and Burstall, 1992).

Dually to colimits, limits represent abstraction, deriving a theory that expresses what theories in a diagram express in common given the theories within which they are used via the morphisms. Colimits exist for all diagrams in a theory category, but limits exist only for certain diagrams, making the situations they express special in some sense. Limits play a fundamental role in an important class of categories called *topoi*, where limits for finite diagrams always exist (Mac Lane and Moerdijk, 1992), as do colimits. Geometric logic arises in a natural way from a certain kind of topos (Vickers, 1992), and the chapter by Vickers, although not making it explicit, draws on a topos of theories which are derived from topoi.

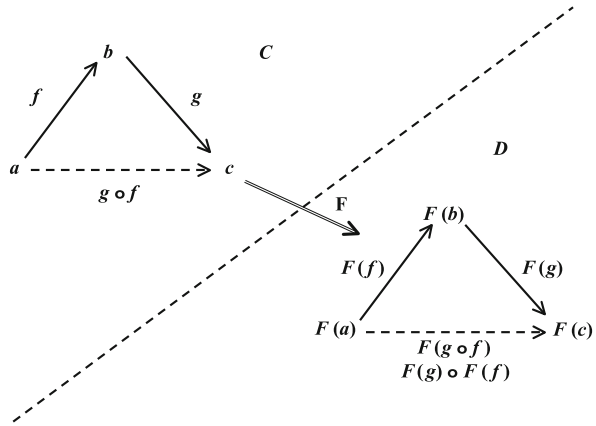
As a final note, theories include not just their presentations, which are shown here, but also their theorems and quantities definable given the presented symbols. The distinction is essential if a strict definition of “presentation morphism” or “specification morphism” is to be enforced as opposed to the more general “theory morphism”, but that is not the case here.

21.4 Structural Mappings

The importance of category theory lies in its ability to formalize the notion that things that differ in matter can have an underlying similarity of form, or structure. A house plan exists as a complex of forms either inscribed in ink on paper or electronically within a computer. The plan can be implemented many times, with variations in the fine details of construction. Each instance of building a house from the plan can be thought of as a mapping from the structure detailed in the architectural plan to a structure made of wood, brick, stone, metal, wallboard, and other materials. The material substances of the plan and the house may differ in each case, but the structure given in the plan is essentially the same in the constructed house. The import of this statement is that the mapping of the details in the plan to the details of the constructed house is more than just an association between elements, associating a particular line in a floor plan with a particular wall of the house, for example: It is also a mapping of structural relationships – for example, two lines that denote walls in a floor plan and meet at a corner must map individually to two walls that meet at a corner in the corresponding position in the house.

A structure-preserving mapping between categories, a *functor* $F: C \longrightarrow D$ with domain category C and codomain category D , associates to each object a of C a unique object $F(a)$ of D and to each morphism $f: a \longrightarrow b$ of C a unique morphism $F(f): F(a) \longrightarrow F(b)$ of D , such that F preserves the compositional structure of C , as follows (see Fig. 21.4). Just this once, let symbols \circ_C and \circ_D denote the composition operations in categories C and D , respectively. Normally, symbols such as \circ are “overloaded” to denote the same kind of thing appearing in different contexts. Now, preservation of structure means that for each composition $g \circ_C f$ in C , F must satisfy the equation $F(g \circ_C f) = F(g) \circ_D F(f)$, and for each identity morphism of C , $F(\text{id}_a) = \text{id}_{F(a)}$. As a consequence, F preserves commutativity, that is, the F -image of a commutative diagram in C is a commutative diagram in D , and isomorphisms are preserved because identities are preserved. These facts, taken

Fig. 21.4 A functor preserves composition, $F(g \circ_C f) = F(g) \circ_D F(f)$



together, mean that any structural constraints expressed in C are translated into D and, hence, F is truly a structure-preserving mapping.

For a simple example, let \mathbf{N}_1^+ be the divisibility category for the positive natural numbers as before. Let \mathbf{N}_{\leq} denote the category whose objects are all the natural numbers $(0, 1, 2, 3, \dots)$ and with a morphism $\leq_{n,m}: n \rightarrow m$ exactly when $n \leq m$. Let the functor $F: \mathbf{N}_1^+ \rightarrow \mathbf{N}_{\leq}$ be defined as follows. The image of each positive natural number n is itself, that is, $F(n) = n$, and the image of $|_{n,m}$ is given by $F(|_{n,m}) = \leq_{n,m}$. That F can be defined is a consequence of the fact that $n | m$ implies $n \leq m$. Notice that the compositional structure of \mathbf{N}_1^+ is appropriately preserved, $F(|_{m,r}) \circ F(|_{n,m}) = \leq_{m,r} \circ \leq_{n,m} = \leq_{n,r} = F(|_{n,r})$. Therefore, F is a functor.

Not only are there structure-preserving mappings between categories, but also structure-preserving relations between the mappings themselves. After building one house of many to be built from a plan, experienced builders can transfer lessons learned from one to the next without having to refer back to the plan for every detail, as long as this transfer is consistent with the plan. Also, there could be changes to details that do not affect the plan structure – finish work, say – requiring only changes made in going from house to house. By analogy – thinking of each house as a “functor” – this transfer of practice must be consistent with the two mappings from the structure specified by the plan to the fitting-together of building materials. The transfer of structural detail from one functor to another is captured in the notion of a *natural transformation* $\alpha: F \rightarrow G$ with domain functor $F: C \rightarrow D$ and codomain functor $G: C \rightarrow D$. This consists of a system of D -morphisms α_a , one for each object a of C , such that the diagram below commutes for each morphism $f: a \rightarrow b$ of C . That is, the morphisms $G(f) \circ \alpha_a: F(a) \rightarrow G(b)$ and $\alpha_b \circ F(f): F(a) \rightarrow G(b)$ are actually one and the same, $G(f) \circ \alpha_a = \alpha_b \circ F(f)$. In a sense, the two functors have their morphism images $F(f): F(a) \rightarrow F(b)$, $G(f): G(a) \rightarrow G(b)$ “stitched together” by other morphisms α_a, α_b existing in D , indexed by the objects of C . Composition of the morphisms along the two paths

leading from one corner $F(a)$ of a commutative square to the opposite corner $G(b)$ yields the same morphism, independently of the path traversed.

$$\begin{array}{ccc}
 F(a) & \xrightarrow{F(f)} & F(b) \\
 \alpha_a \downarrow & & \downarrow \alpha_b \\
 G(a) & \xrightarrow{G(f)} & G(b)
 \end{array}$$

Functors and natural transformations underlie the constructs of computer science (Pierce, 1991; Walters, 1991; Crole, 1993). Pierce describes the functor $\mathbf{List} : \mathbf{Set} \rightarrow \mathbf{Mon}$ that maps each set X to the monoid $(\mathbf{List}_X, *, [])$ where \mathbf{List}_X is the set of finite strings, or lists, that can be formed from the elements of X ; thus, $\mathbf{List}(X) = (\mathbf{List}_X, *, [])$. A monoid is an algebra (A, σ, e) where A is a set, $\sigma : A \times A \rightarrow A$ is a binary operation on A mapping pairs of elements to elements, $z = \sigma(x, y)$ where $x, y, z \in A$, and e , where $e \in A$, is an identity element for σ . Binary operations are normally written in infix form, so one writes $\sigma(x, y)$ as $x \sigma y$. The element e has the property that $x \sigma e = x$ and $e \sigma x = x$ for all $x \in A$. Pairs of equations like this with a common left or right hand side are condensed to the form $x \sigma e = x = e \sigma x$. The monoid operation σ is also associative, $x \sigma (y \sigma z) = (x \sigma y) \sigma z$. The monoid operation for $(\mathbf{List}_X, *, [])$ is list concatenation $[x_1, x_2, \dots, x_n] * [y_1, y_2, \dots, y_m] = [x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m]$, where $x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m \in X$; e is the null list $[],$ where for example, $[x_1, x_2, \dots, x_n] * [] = [x_1, x_2, \dots, x_n]$. A monoid homomorphism $h : (A, \sigma, e) \rightarrow (A', \sigma', e')$ is a function $h : A \rightarrow A'$ that preserves the monoid operation, $h(x \sigma y) = h(x) \sigma' h(y)$ and $h(e) = e'$. Notice that h has been used in two ways here, to denote both a homomorphism and its underlying set function. This is common practice among algebraists, and is used by mathematicians in other fields as well for economy of notation. The objects of the category \mathbf{Mon} are monoids and its morphisms are monoid homomorphisms.

The fact that \mathbf{List} is a functor, and not just a function from sets to monoids, can be seen by applying it to any function on list elements (which are elements of sets) as the familiar `maplist` operator: If X and Y are sets (objects of \mathbf{Set}) and $f : X \rightarrow Y$ is a function (a morphism of \mathbf{Set}), then

$$\begin{aligned}
 \mathbf{List}(f)([x_1, x_2, \dots, x_n]) &= [f(x_1), f(x_2), \dots, f(x_n)] \\
 &= \mathbf{maplist}(f)([x_1, x_2, \dots, x_n]).
 \end{aligned}$$

This shows the underlying set function of $\mathbf{List}(f)$ with domain \mathbf{List}_X and codomain \mathbf{List}_Y . It is easily shown that $\mathbf{List}(f)$ is a monoid homomorphism

$\text{List}(f) : (\text{List}_X, *, []) \longrightarrow (\text{List}_Y, *, [])$ (hereafter written $\text{List}(f) : \text{List}(X) \longrightarrow \text{List}(Y)$), that is, that

$$\begin{aligned} \text{List}(f)([x_1, x_2, \dots, x_n] * [y_1, y_2, \dots, y_m]) &= \\ \text{List}(f)([x_1, x_2, \dots, x_n]) * \text{List}(f)([y_1, y_2, \dots, y_m]) \end{aligned}$$

and $\text{List}(f)([]) = []$. For our purpose, it is more instructive to show that List itself is a functor. Note that if $f: X \longrightarrow Y$ and $g: Y \longrightarrow Z$ are **Set** morphisms, then for a monoid element $[x_1, x_2, \dots, x_n]$ of List_X ,

$$\begin{aligned} \text{List}(g \circ f)([x_1, x_2, \dots, x_n]) &= [(g \circ f)(x_1), (g \circ f)(x_2), \dots, (g \circ f)(x_n)] \\ &= [g(f(x_1)), g(f(x_2)), \dots, g(f(x_n))] \\ &= \text{List}(g)([f(x_1), f(x_2), \dots, f(x_n)]) \\ &= \text{List}(g)(\text{List}(f)([x_1, x_2, \dots, x_n])) \\ &= (\text{List}(g) \circ \text{List}(f))([x_1, x_2, \dots, x_n]), \end{aligned}$$

and therefore $\text{List}(g \circ f) = \text{List}(g) \circ \text{List}(f)$. Finally,

$$\begin{aligned} \text{List}(\text{id}_X)([x_1, x_2, \dots, x_n]) &= [\text{id}_X(x_1), \text{id}_X(x_2), \dots, \text{id}_X(x_n)] \\ &= [x_1, x_2, \dots, x_n] \\ &= \text{id}_{\text{List}(X)}([x_1, x_2, \dots, x_n]), \end{aligned}$$

so $\text{List}(\text{id}_X) = \text{id}_{\text{List}(X)}$.

An example of a natural transformation is given by the reversal operation on lists, $\text{rev} : \text{List} \longrightarrow \text{List}$, whose component for each object X of **Set** is the monoid homomorphism $\text{rev}_X : \text{List}(X) \longrightarrow \text{List}(X)$, defined on elements $[x_1, x_2, \dots, x_n] \in \text{List}_X$ by $\text{rev}_X([x_1, x_2, \dots, x_n]) = [x_n, \dots, x_2, x_1]$. Then, for any **Set** morphism $f: X \longrightarrow Y$,

$$\begin{aligned} (\text{List}(f) \circ \text{rev}_X)([x_1, x_2, \dots, x_n]) &= \text{List}(f)(\text{rev}_X([x_1, x_2, \dots, x_n])) \\ &= \text{List}(f)([x_n, \dots, x_2, x_1]) \\ &= [f(x_n), \dots, f(x_2), f(x_1)] \\ &= \text{rev}_Y([f(x_1), f(x_2), \dots, f(x_n)]) \\ &= \text{rev}_Y(\text{List}(f)([x_1, x_2, \dots, x_n])) \\ &= (\text{rev}_Y \circ \text{List}(f))([x_1, x_2, \dots, x_n]). \end{aligned}$$

Then $\text{rev}_X : \text{List}(X) \longrightarrow \text{List}(X)$ and $\text{rev}_Y : \text{List}(Y) \longrightarrow \text{List}(Y)$ commute with the List -image of any $f: X \longrightarrow Y$, and therefore, rev is a natural transformation mapping the functor List to itself.

21.5 Categories of Categories, Functors, and Natural Transformations

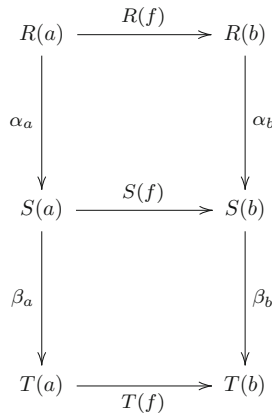
Applications of category theory inevitably involve functors, many with special properties. Often, in fact, they involve categories of categories and functors, and categories of functors and natural transformations. Two functors $F: C \longrightarrow D$

and $G : D \rightarrow E$, where C, D , and E are categories, have a composition $G \circ F : C \rightarrow E$, defined as follows using two C -morphisms $f : a \rightarrow b$ and $g : b \rightarrow c$ of C :

$$(G \circ F)(g \circ_C f) = G(F(g \circ_C f)) = G[F(g) \circ_D F(f)] = G(F(g)) \circ_E G(F(f))$$

Natural transformations have two (quite natural!) notions of composition. First, consider three functors having the same domain and codomain category, $R, S, T : C \rightarrow D$, and two natural transformations $\alpha : R \rightarrow S$ and $\beta : S \rightarrow T$. The *vertical composition* of α and β , $\beta \cdot \alpha : R \rightarrow T$, is defined simply by forming the compositions of the corresponding components $\alpha_a : R(a) \rightarrow S(a)$ and $\beta_a : S(a) \rightarrow T(a)$, $(\beta \cdot \alpha)_a = \beta_a \circ \alpha_a$, which after all are just morphisms of D , the common codomain category of the three functors (the reason for using \cdot rather than \circ to denote vertical composition will be made clear presently). For any pair a, b of objects of C , the resulting commutative diagram for $\beta \cdot \alpha$ is obtained by “pasting together” the commutative diagrams for α and β along their common side as shown below. Commutative diagrams with a common side can always be merged to form a larger diagram. In this case, simply observe that

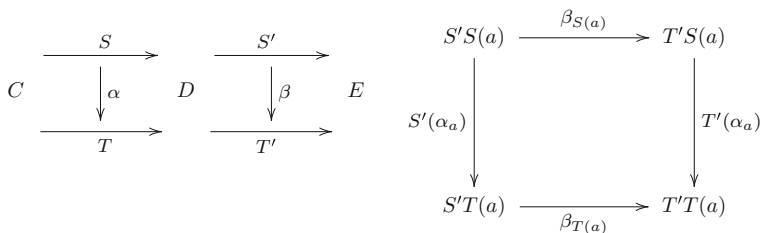
$$\begin{aligned} (\beta_b \circ \alpha_b) \circ R(f) &= \beta_b \circ (\alpha_b \circ R(f)) \\ &= \beta_b \circ (S(f) \circ \alpha_a) \\ &= (\beta_b \circ S(f)) \circ \alpha_a \\ &= (T(f) \circ \beta_a) \circ \alpha_a \\ &= T(f) \circ (\beta_a \circ \alpha_a) \end{aligned}$$



The compositions of functors and natural transformations as defined are both easily shown to be associative. In addition, for each category C there is an identity functor $\text{id}_C : C \rightarrow C$ mapping its objects and morphisms to themselves, and also

an identity natural transformation $\text{id}_F: F \rightarrow F$ for each functor $F: C \rightarrow D$ which has the identity morphisms $\text{id}_{F(a)}: F(a) \rightarrow F(a)$ of D as components. This leads to the notion of categories in which the objects are categories and the morphisms are functors, and also the notion of functor categories D^C for categories C and D , which have functors $F: C \rightarrow D$ as objects and natural transformations with vertical composition as the morphisms.

In fact, many applications involve both kinds of natural transformation composition. Consider two pairs of functors $S, T: C \rightarrow D$ and $S', T': D \rightarrow E$ and two compositions $S' \circ S, T' \circ T: C \rightarrow E$. The *horizontal composition* of two natural transformations $\alpha: S \rightarrow T$ and $\beta: S' \rightarrow T'$ is a natural transformation $\beta \circ \alpha: S' \circ S \rightarrow T' \circ T$ whose components $(\beta \circ \alpha)_a: (S' \circ S)(a) \rightarrow (T' \circ T)(a)$ for each object a of C can be obtained via either of the two compositions $T'(\alpha_a) \circ \beta_{S(a)}$ or $\beta_{T(a)} \circ S'(\alpha_a)$. This is because the diagram on the right below commutes, since β is natural for morphisms of D , and, in particular, for the morphisms α_a . Actually proving that $\beta \circ \alpha$ is natural involves more diagrams but is not difficult; see the classic book (Mac Lane, 1971). The diagram on the left shows the overall situation involving the three categories, two pairs of functors, and two natural transformations.

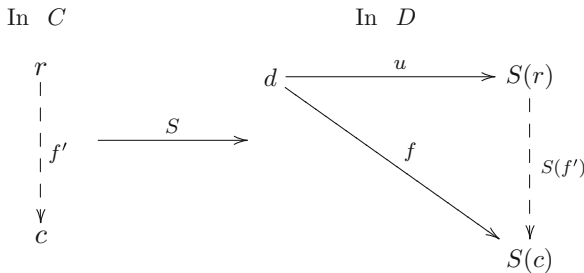


Oddly enough, functors can be composed with natural transformations. This is because a functor T can be regarded as the natural transformation $\text{id}_T: T \rightarrow T$. Letting $S = T$ in the above right-hand diagram and $\alpha = \text{id}_T$, and thinking of T also as id_T , one defines a composition which is usually denoted βT by the equations $\beta T = \beta \circ T = \beta \circ \text{id}_T$. Making the appropriate substitutions in the diagram shows that a typical component of βT is $(\beta T)_a = \beta_{T(a)}$. On the other hand, letting, instead, $S' = T'$ and $\beta = \text{id}_{T'}$, one obtains $T'\alpha$ by the equations $T'\alpha = T' \circ \alpha = \text{id}_{T'} \circ \alpha$, with typical component $(T'\alpha)_a = T'(\alpha_a)$.

Having both vertical and horizontal composition leads to categories having a two-dimensional structure. Because the identity natural transformations are the same for both compositions, these are called 2-categories. Both kinds of composition can be applied to the simpler functor categories D^C with certain restrictions. The case $E = D = C$ yields a 2-category C^C of endofunctors $T: C \rightarrow C$ which includes compositions of the form αT and $T\alpha$. Mac Lane's classic is probably the best reference for vertical and horizontal composition, as with most of the concepts discussed in this section.

21.6 Universal Arrows and Adjunctions

The images of the objects and morphisms of a category C mapped into a category D by a functor $S: C \rightarrow D$ typically cover only a part of D . In many cases, S is accompanied by morphisms in D that act as prime factors in connecting an arbitrary object d of D with the objects $S(c)$. Such morphisms are called *universal arrows*. Formally, a *universal arrow from d to S* is a pair (r, u) , where r is an object of C and u is a morphism $u: d \rightarrow S(r)$ of D , such that for every D -morphism of the form $f: d \rightarrow S(c)$, there is a unique C -morphism $f': r \rightarrow c$ in C with $f = S(f') \circ u$. The picture for this involves a C -arrow and a commutative triangle in D :



The dual notion of a *universal arrow from S to d* is obtained by reversing the arrows u, f , and f' and the order of composition but leaving everything else unchanged (in particular, notice that dualizing does not mean reversing the functor). In either case, the occurrence of universal arrows entails a special relationship between C, D , and S .

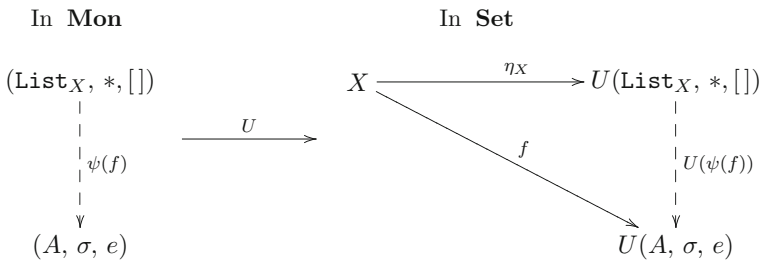
To illustrate, let us make further use of the example involving the functor $\text{List}: \mathbf{Set} \rightarrow \mathbf{Mon}$, as does Pierce (1991). First, recall that the monoid homomorphism symbol $\text{List}(f)$, where $f: X \rightarrow Y$ is a set function (morphism), was overloaded so that it meant both the homomorphism $\text{List}(f): (\text{List}_X, *, []) \rightarrow (\text{List}_Y, *, [])$ and its underlying set function $\text{List}(f): \text{List}_X \rightarrow \text{List}_Y$, as is common practice. This practice will continue here where needed for economy of notation, however, this example will now involve a functor U that will make the distinction between monoids and their underlying sets explicit. For an arbitrary monoid (A, σ, e) , let $U(A, \sigma, e)$ denote the operation of extracting its underlying set, so that $A = U(A, \sigma, e)$; so, now, $\text{List}_X = U(\text{List}_X, *, [])$. Likewise, the underlying function of a monoid homomorphism $h: (A, \sigma, e) \rightarrow (A', \sigma', e')$ can be expressed as either of the equivalent forms $U(h): A \rightarrow A'$ and $U(h): U(A, \sigma, e) \rightarrow U(A', \sigma', e')$. Similarly, applying U to $\text{List}(f)$ yields $U(\text{List}(f)): U(\text{List}_X, *, []) \rightarrow U(\text{List}_Y, *, [])$, which is another way of expressing the set function $U(\text{List}(f)): \text{List}_X \rightarrow \text{List}_Y$.

Now let X be an arbitrary object of \mathbf{Set} and let $\eta_X: X \rightarrow \text{List}_X$ be the function that maps each element $x \in X$ to the singleton list $[x]$ in the set of lists List_X , which is just another name for $U(\text{List}_X, *, [])$. Let (A, σ, e) be an arbitrary

monoid, and let $f: X \rightarrow U(A, \sigma, e)$ be a function (note carefully: f is an arrow in **Set**). Stated another way, $f: X \rightarrow A$, so for each $x \in X$, there must be a unique monoid element $f(x) \in A$. Correspondingly, there is a monoid homomorphism (note: an arrow in **Mon**) $\psi(f): \text{List}(X) \rightarrow (A, \sigma, e)$ obtained as follows: Let $\psi(f)([]) = e$, $\psi(f)([x]) = f(x)$ where $x \in X$, and $\psi(f)([x_1, x_2, \dots, x_n]) = f(x_1) \sigma f(x_2) \sigma \dots \sigma f(x_n)$. This is easily shown to define a homomorphism: Simply note that

$$\begin{aligned} \psi(f)([x_1, x_2, \dots, x_n] * [y_1, y_2, \dots, y_m]) &= \psi(f)([x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m]) \\ &= f(x_1) \sigma f(x_2) \sigma \dots \sigma f(x_n) \sigma f(y_1) \sigma f(y_2) \sigma \dots \sigma f(y_m) \\ &= (f(x_1) \sigma f(x_2) \sigma \dots \sigma f(x_n)) \sigma (f(y_1) \sigma f(y_2) \sigma \dots \sigma f(y_m)) \\ &= \psi(f)([x_1, x_2, \dots, x_n]) \sigma \psi(f)([y_1, y_2, \dots, y_m]). \end{aligned}$$

The underlying set function of the homomorphism $\psi(f): \text{List}(X) \rightarrow (A, \sigma, e)$ can now be extracted, yielding a **Set** morphism $U(\psi(f)): U(\text{List}(X)) \rightarrow U(A, \sigma, e)$. Then for $x \in X$, $(U(\psi(f)) \circ \eta_X)(x) = U(\psi(f))(\eta_X(x)) = U(\psi(f))([x]) = f(x)$, since $U(\psi(f))$ simply extracts the underlying set function f of $\psi(f)$. Therefore, η_X is a universal arrow from X to U . The situation is pictured as follows with the monoids displayed in full:



But there is more to this example, this time in **Mon**: For any monoid (A, σ, e) , the universal arrow from List to (A, σ, e) is the arrow $\xi_{(A, \sigma, e)}: \text{List}(U(A, \sigma, e)) \rightarrow (A, \sigma, e)$, or $\xi_{(A, \sigma, e)}: (\text{List}_A, *, []) \rightarrow (A, \sigma, e)$, whose underlying function is defined on $[a_1, a_2, \dots, a_n] \in \text{List}_A$ by $\xi_{(A, \sigma, e)}([a_1, a_2, \dots, a_n]) = a_1 \sigma a_2 \sigma \dots \sigma a_n$. The proof that it is a homomorphism is identical with that for the homomorphism $\psi(f)$. To see that $\xi_{(A, \sigma, e)}$ is universal as stated, let $h: \text{List}(X) \rightarrow (A, \sigma, e)$ be an arbitrary homomorphism given some object X of **Set**. Because h is a homomorphism,

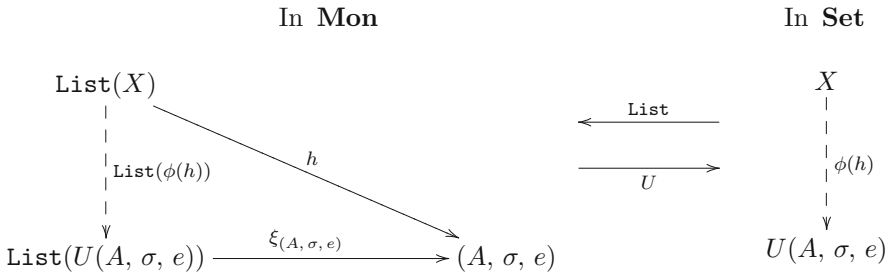
$$\begin{aligned} h([x_1, x_2, \dots, x_n]) &= h([x_1] * [a_2] * \dots * [a_n]) \\ &= h([x_1]) \sigma h([x_2]) \sigma \dots \sigma h([x_n]). \end{aligned}$$

Now, for each singleton $[x] \in \text{List}_X$, $h([x])$ is an element of the underlying set A , $h([x]) \in A$. This says that there is an arrow $\phi(h): X \rightarrow A = U(A, \sigma, e)$ in **Set**

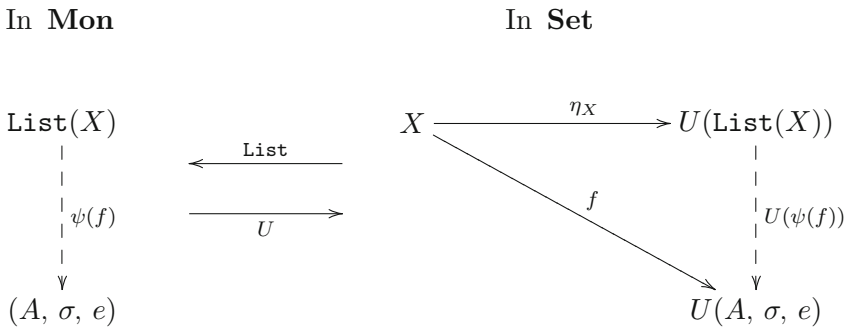
given by $\phi(h)(x) = h([x])$. Then, $\text{List}(\phi(h)): \text{List}(X) \rightarrow \text{List}(U(A, \sigma, e))$ is given by $\text{List}(\phi(h))([x_1, x_2, \dots, x_n]) = [\phi(h)(x_1), \phi(h)(x_2), \dots, \phi(h)(x_n)]$. Then

$$\begin{aligned} h([x_1, x_2, \dots, x_n]) &= h([x_1]) \sigma h([x_2]) \sigma \dots \sigma h([x_n]) \\ &= \phi(h)(x_1) \sigma \phi(h)(x_2) \sigma \dots \sigma \phi(h)(x_n) \\ &= \xi_{(A, \sigma, e)}([\phi(h)(x_1), \phi(h)(x_2), \dots, \phi(h)(x_n)]) \\ &= \xi_{(A, \sigma, e)}(\text{List}(\phi(h))([x_1, x_2, \dots, x_n])) \\ &= (\xi_{(A, \sigma, e)} \circ \text{List}(\phi(h)))([x_1, x_2, \dots, x_n]). \end{aligned}$$

This shows that $h = \xi_{(A, \sigma, e)} \circ \text{List}(\phi(h))$. Therefore, $\xi_{(A, \sigma, e)}$ is a universal arrow from List to (A, σ, e) . The situation is pictured as follows:



Let us replay the picture involving the universal arrow η_X , where the picture below has been modified so that the actions of *both* functors U and List are explicitly shown, as was done for the picture involving $\xi_{(A, \sigma, e)}$:



Now notice that there is evidently a function η assigning a universal arrow η_X to each object X of **Set** and also a function ξ assigning a universal arrow $\xi_{(A, \sigma, e)}$ to each object (A, σ, e) of **Mon**. In fact, replacing X in the picture for η by its identical functor image $\text{id}_{\text{Set}}(X)$ is suggestive, because $\eta: \text{id}_{\text{Set}} \rightarrow U \circ \text{List}$ is a

natural transformation; simply notice that the following diagram in **Set** commutes, where $f: X \rightarrow Y$, shown as $\text{id}_{\text{Set}}(f): \text{id}_{\text{Set}}(X) \rightarrow \text{id}_{\text{Set}}(Y)$, is a **Set** morphism:

$$\begin{array}{ccc}
 \text{id}_{\text{Set}}(X) & \xrightarrow{\eta_X} & (U \circ \text{List})(X) \\
 \text{id}_{\text{Set}}(f) \downarrow & & \downarrow U(\psi(f)) \\
 \text{id}_{\text{Set}}(Y) & \xrightarrow{\eta_Y} & (U \circ \text{List})(Y)
 \end{array}$$

(The sense of this diagram has been rotated ninety degrees compared to previously-shown diagrams for natural transformations.) A similar diagram can be constructed for the natural transformation $\xi: \text{List} \circ U \rightarrow \text{id}_{\text{Mon}}$. It can also be shown that ϕ and ψ are natural transformations on certain categories, such that there is a one-to-one correspondence between the pairs of arrows h and $\phi(h)$, f and $\psi(f)$ used to illustrate the example.

There are yet more aspects to this example, and there is a resulting symmetry between the two systems of universal arrows and **Set**, **Mon**, U and List . This kind of situation occurs very often in category theory and is so important that it is given a name: *adjunction*. The functor List is called the *left adjoint* (or *left adjoint*) of the functor U , and U is called the *right adjoint* (or *right adjoint*) of List . Adjunctions are central to mathematical semantics, and it is with regret that this chapter must end without further exploration of this concept. Even more regrettably, no attention has been given to *topoi*, another concept essential for semantics. The chapters by Johnson and Rosebrugh and Kent do explore adjunctions, and Vickers explores geometric theories, which are derived from *topoi*.

References

- Adamek, J., H. Herrlich, and G. Strecker. 1990. *Abstract and concrete categories*. New York, NY: Cambridge University Press.
- Baianu, I.C. 1988. Computer models and automata theory in biology and medicine. In *Mathematical Models in Medicine*, ed. M. Witten, vol. 7, 1513–1577. New York, NY: Pergamon Press.
- Baianu, I.C., R. Brown, and J.F. Glazebrook. 2007. Categorical ontology of complex spacetime structures: The emergence of life and human consciousness. *Axiomathes* 17:3–4, 209–222.
- Burstall, R., and J. Goguen. 1977. Putting theories together to make specifications. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Department of Computer Science, Carnegie-Mellon University, ed. R. Reddy, 1045–1058.
- Burstall, R., and J. Goguen. 1980. The semantics of Clear, a specification language. In *Proceedings, 1979 Copenhagen Winter School on Abstract Software Specification*, LNCS, ed. D. Bjorner, vol. 86, 292–332. Springer.
- Colomb, R., C.N.G. Dampney, and M. Johnson. 2001. The use of category-theoretic fibration as an abstraction mechanism in information systems. *Acta Informatica* 38(1):1–44.
- Crole, R.L. 1993. *Categories for types*. Cambridge, MA: Cambridge University Press.

- Dampney, C.N.G., M. Johnson, and R. Rosebrugh. 2001. View updates in a semantic data modelling paradigm. In *Proceedings of the 12th Australasian Database Conference ADC2001*, 29–36. IEEE Press.
- Ehresmann, A.C., and J.-P. Vanbreemsch. 1997. Information processing and symmetrybreaking in memory evolutive systems. *Biosystems* 43:25–40.
- Goguen, J. 1973. Categorical foundations for general systems theory. In *Advances in cybernetics and systems research*, eds. F. Pichler, and R. Trappl, 121–130. London: Transcripta Books.
- Goguen, J.A., and R.M. Burstall. 1984. Some fundamental algebraic tools for the semantics of computation – Part 1: Comma categories, colimits, signatures and theories. *Theoretical Computer Science* 31(1,2):175–209.
- Goguen, J.A., and R.M. Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery* 39(1): 95–146.
- Gust, H., and K.-U. Kühnberger. 2005. Learning symbolic inferences with neural networks. In *Proceedings of the XXVII Annual Conference of the Cognitive Science Society (CogSci2005)*. Cognitive Science Society.
- Healy, M. J., and T. P. Caudell. 2006. Ontologies and worlds in category theory: Implications for neural systems. *Axiomathes* 16(1–2):165–214.
- Healy, M. J., and K.E. Williamson. 2000. Applying category theory to derive engineering software from encoded knowledge. In *Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000, Iowa City, Iowa, USA, Proceedings, LNCS 1816*, ed. T. Rus, 484–498, Springer-Verlag.
- Johnson, M., and R. Rosebrugh. 2001. View updatability based on the models of a formal specification. In *Proceedings of the 10th International Symposium of Formal Methods Europe (FME 2001, Berlin, March 12–16): Formal methods for increasing software productivity*, 534–549.
- Jullig, R., and Y.V.Srinivas. 1993. Diagrams for software synthesis. In *Proceedings of KBSE '93: The 8th Knowledge-Based Software Engineering Conference*, 10–19, IEEE Computer Society Press.
- Kent, R.E. 2003. The IFF foundation for ontological knowledge organization. In *Knowledge Organization and Classification in International Information Retrieval*, 187–204.
- Kernighan, B.W., and D.M. Ritchie. 1988. *The C programming language*, 2nd edition. Englewood Cliffs, NJ: Prentice Hall.
- Lawvere, F.W. 1963. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences* 50:869–873.
- Lawvere, F.W., and S.H. Schanuel. 1995. *Conceptual mathematics: A first introduction to categories*. Cambridge, NY: Cambridge University Press.
- Mac Lane, S. 1971. *Categories for the working mathematician*. New York, NY: Springer.
- Mac Lane, S., and I. Moerdijk. 1992. *Sheaves in geometry and logic: A first introduction to topos theory*. New York, NY: Springer.
- Medin, D.L. 1989. Concepts and conceptual structure. *American Psychologist* 44(12):1461–1481.
- J. Meseguer. 1989. General logics. In *Logic Colloquium '87*, eds. H.-D. E. et al., 275–329. Amsterdam: Science Publishers B.V. (North-Holland).
- Pierce, B.C. 1991. *Basic category theory for computer scientists*. Cambridge, MA: MIT Press.
- Rosen, R. 1958. The representation of biological systems from the standpoint of the theory of categories. *Bulletin Of Mathematical Biophysics* 20:317–341.
- Schorlemmer, M., and Y. Kalfoglou. 2005. Progressive ontology alignment for meaning coordination: An information-theoretic foundation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, ed. F. Dignum, 737–744. Netherland: ACM Press.
- Uschold, M., M. Healy, K. Williamson, P. Clark, and S. Woods. 1998. Ontology reuse and application. In *International Conference on Formal Ontology in Information Systems (FOIS '98)*, Trento, Italy, Proceedings.

- Vickers, S. 1992. Geometric theories and databases. In *Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham 1991*. eds. M.P. Fourman, P. T. Johnstone, and A.M. Pitts, 288–314. New York, NY: Cambridge University Press.
- Walters, R.F.C. (ed.). (1991). *Categories and computer science (No. 28)*. Cambridge: Cambridge University Press.
- Williamson, K. E., and M.J. Healy. 2000. Deriving engineering software from requirements. *Journal of Intelligent Manufacturing*, 11(1):3–28.

Chapter 22

Issues of Logic, Algebra and Topology in Ontology

Steven Vickers

22.1 Introduction

In my book “Topology via Logic” (Vickers, 1989) I motivated the application of topology to computer science by presenting topologies as *observational* accounts of (e.g.) computer programs, with open sets representing observable properties and the axioms of topology reflecting a logic of observations. This developed ideas of Smyth and Abramsky, and seemed to provide a useful explanation of how topologies were used in denotational semantics.

The logic involved (*geometric* logic) is well known in topos theory, and has a predicate form going somewhat beyond the propositional logic of my book. This chapter is presented as an ontological examination of the logic, developing the observational ideas in my book. (To some extent these were already sketched in Vickers (1992).)

The title mentions logic, algebra and topology, which together cover vast parts of mathematics, and indeed geometric logic does have deep and subtle connections with all those. It should, however, be clear that a short chapter such as this cannot give a comprehensive survey of the connections between those parts of mathematics and ontology. Instead, we shall focus on geometric logic as a case study for an ontological examination, and briefly mention its connections with those broader fields. The logic brings together topology and algebra in some rather remarkable ways, and has an inherent continuity.

We shall describe in fair detail the presentation of geometric logic as given in Johnstone (2002b), and from this technical point of view there is little new. However, we shall also use that as the basis for a novel ontological discussion, with particular emphasis on the question *What is the ontological commitment of the logic?* The logic in itself avoids certain ontological problems with classical logic, and that is our primary reason for choosing it as a case study. However, we shall also see that the mode of presentation in Johnstone (2002b), using sequents rather than sentences, in

S. Vickers (✉)
School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK
e-mail: s.j.vickers@cs.bham.ac.uk

itself facilitates the analysis. This is because it makes a clear distinction between *formulae* and *axioms*, and that reflects an ontological distinction between *observations* and *hypotheses*.

Our naive view of ontological questions is that they concern the connection between symbols and the world. Russell (1945, Chap. XVIII, Knowledge and Perception in Plato) says that for a symbolic formula to *exist* we must demonstrate an instance of it in the world. If we say “lions exist, but unicorns don’t”, then to prove our point with regard to lions, we go to the zoo and identify something that we can agree is an instance of “lion”. Of course, this presupposes an understanding of what would constitute such an instance, but we think the scenario is a plausible one. Another day we instead go looking for a unicorn and this time our outing is less successful. Despite our seeing a rhinoceros, a goat with a horn missing, a plastic toy unicorn and a royal coat of arms, none of these seems truly satisfactory and we return home still not knowing whether unicorns exist. Nonetheless, we can agree it was still worth trying.

This ontological connection between symbols and world is clearly not in itself part of formal logic. Nonetheless, we shall argue informally how formal features of the logic can make it easier to analyse the informal connection.

Our main thesis has two parts.

First, we shall be arguing that a formal logic (with connectives and rules of inference) carries a certain *ontological commitment* to how it could be interpreted “in the real world”. Classical first-order logic uses various symbols that on the face of it have a straightforward relationship with concepts of everyday life: \wedge (*conjunction*) means “and”, \vee (*disjunction*) means “or”, \neg (*negation*) means “not”, \rightarrow (*implication*) means “implies”, \forall (*universal quantification*) means “for all” and \exists (*existential quantification*) means “for some”. However, we shall argue that the way classical logic deals with these adds up to a very strong ontological commitment that could be problematic in reality. Specifically, negation (\neg), implication (\rightarrow) and universal quantification (\forall) cannot be expected to have a uniform interpretation on the same level as conjunction (\wedge) and disjunction (\vee). This suggests a need to consider other less standard logics to describe “the real world”.

In fact, even in formal mathematics this need can make itself felt. The strong ontological commitment of classical logic can be sustained in formal mathematics (in particular, in set theory), but only because that ontological commitment is already built in to the way set theory is formalized. In other settings it may cause problems. One example we shall briefly mention later (Section 22.6.1) is *sheaves over a topological space* X . The geometric logic that we shall describe was invented to be used with sheaves, and indeed for more general contexts known as *toposes*. That background is not needed here, but it means there is a well established mathematical setting in which geometric logic can be interpreted.

The second part of our thesis is that it is fruitful to examine the ontological commitment of geometric logic, and explore how it might be interpreted in “the real world”. We do not claim that it is *the* right logic to use, but it avoids the more immediate problems of classical logic.

In Section 22.6 we shall touch on the relationship with categories, a pervasive theme in other chapters in this collection.

22.2 Ingredients of Logic

We first review the ingredients of logic. We shall adopt a sequent approach, specifically that set out in Johnstone (2002b). Technically, this is all well established (apart from some of the notation); what is new is the ontological discussion. We include enough in this Section to show how the sequent approach facilitates a more careful ontological analysis. In Section 22.3 we shall discuss in more detail geometric logic and an ontology for it.

A *many-sorted, first-order signature* has a set of sorts, a set of predicate symbols, and a set of function symbols. Each predicate or function symbol has an *arity* stipulating the number and sorts of its arguments, and (for a function) the sort of its result. A predicate symbol with no arguments is *propositional*, while a function with no arguments is a *constant*. We shall express the arities of predicates and functions thus:

$$\begin{array}{ll} P \subseteq A_1, \dots, A_n & \text{(for a predicate)} \\ P \subseteq 1 & \text{(for a proposition)} \\ f : A_1, \dots, A_n \rightarrow B & \text{(for a function)} \\ c : B & \text{(for a constant)} \end{array}$$

These symbols in the signature are *extra-logical* – outside the logic. They are meaningless until interpreted. Since the nature of the interpretation will be very important in our ontological discussion, we shall introduce a non-standard notation that makes the interpretation quite explicit. Suppose we have an interpretation that we call M . Mathematically, M must interpret each sort A as a *set*, the *carrier* for A , which we shall write as $\{M|A\}$.

Note that we do *not* presume that $\{M|A\}$ has any elements. The normal account of classical logic requires each carrier to be non-empty, but this is actually a big ontological commitment.

A function symbol $f : A_1, \dots, A_n \rightarrow B$ is used to construct terms of sort B by applying f to n arguments of sorts A_1, \dots, A_n . In M , therefore, the interpretation of f should tell us how, if we are given arguments in the form of values $a_i \in \{M|A_i\}$, there is then a corresponding result $f(a_1, \dots, a_n) \in \{M|B\}$. Hence f is interpreted as a function from the cartesian product $\prod_{i=1}^n \{M|A_i\}$ to $\{M|B\}$. To simplify notation we shall write $\{M|A_1, \dots, A_n\}$ for that cartesian product, and to simplify it further we shall often use vector notation $\{M|\vec{A}\}$. Then the vector $(a_1, \dots, a_n) = \vec{a} \in \{M|\vec{A}\}$ and the interpretation of f is as a function

$$\{M|f\} : \{M|\vec{A}\} \rightarrow \{M|B\}.$$

A constant $c : B$ is a function in the special case of having no arguments and so is interpreted as an element

$$\{M|c\} \in \{M|B\}.$$

For a predicate $P \subseteq A_1, \dots, A_n$, the interpretation needs to say for which argument tuples $\vec{a} \in \vec{A}$ the predicate $P(a_1, \dots, a_n)$ is true. Hence it is equivalent to specifying a subset of the set of all tuples (of the right sorts):

$$\{M|P\} \subseteq \{M|\vec{A}\}.$$

Just as with constants, a proposition $P \subseteq 1$ is a predicate in the special case of having no arguments. This is, as one would expect, interpreted as a truth value. However, we can also see this as a special case of predicates with arguments. If the vector \vec{A} is empty – its length is zero – then for $\{M|\vec{A}\}$ we are looking for the “cartesian product of no sets”. The most natural interpretation of this is the 1-element set whose only element is the empty (zero length) vector ε (say). A subset $\{M|P\} \subseteq \{\varepsilon\}$ is determined solely by the truth value of $\varepsilon \in \{M|P\}$. If the truth value is **true** then $\{M|P\} = \{\varepsilon\}$, while if the truth value is **false** then $\{M|P\} = \emptyset$. Any one-element set will do for this purpose, which is why we write $P \subseteq 1$ to say that P is a propositional symbol.

Once the signature is given, *terms* can be built up in the usual way. A term will usually contain variables, and if \vec{x} is a list of distinct variables x_i , each with a stipulated sort $\sigma(x_i)$, then we say that a term t is *in context* \vec{x} if all its variables are amongst the x_i s. We also say that $(\vec{x}.t)$ is a term in context.

- Each variable x is a term of sort $\sigma(x)$.
- Suppose $f : A_1, \dots, A_n \rightarrow B$ is a function symbol in the signature, and for each i ($1 \leq i \leq n$), t_i is a term in context \vec{x} of sort A_i . Then $f(t_1, \dots, t_n)$ (or $f(\vec{t})$) is a term in context \vec{x} of sort B .

If an interpretation M is given for the signature, then it extends to all terms. Consider a term in context $(\vec{x}.t)$ of sort $\sigma(t)$. If values $\vec{a} \in \{M|\sigma(\vec{x})\}$ are given, then they can be substituted for the variables \vec{x} and then the whole expression can be evaluated in an obvious way to get an element of $\{M|\sigma(t)\}$. Thus the term in context is interpreted as a function

$$\{M|\vec{x}.t\} : \{M|\sigma(\vec{x})\} \rightarrow \{M|\sigma(t)\}.$$

More systematically, we can say how to evaluate this using the structure of t .

The simplest case is when t is just one of the variables, say x_i . Then the function is the projection function

$$\{M|\vec{x}.x_i\}(\vec{a}) = a_i.$$

Note something important here. The context has variables that are not used in the term, but they still influence the way the term is interpreted. We cannot define the interpretation of the term x_i without also knowing what context \vec{x} it is taken to be in.

Now suppose we have a term of the form $f(\vec{t})$ where each t_i is a term in context \vec{x} . Once we have calculated $\{M|\vec{x}.t_i\}(\vec{a})$ for these subterms, then we can say

$$\{M|\vec{x}.f(\vec{t})\}(\vec{a}) = \{M|f\}(\{M|\vec{x}.t_1\}(\vec{a}), \dots, \{M|\vec{x}.t_n\}(\vec{a})).$$

In the special case where the context \vec{x} is empty (so the term is *closed* – it has no variables), we use exactly the same procedure but bearing in mind that the vector \vec{a} is empty (ε). This gives us a function from $\{\varepsilon\}$ to $\{M|\sigma(t)\}$, which is equivalent to picking out an element $\{M|\varepsilon.f(\vec{t})\} \in \{M|\sigma(f(\vec{t}))\}$.

Next we look at formulae in context. We start by describing all the ways that formulae can be constructed in classical first-order predicate logic. However, we shall later retreat from this in the face of problems that are essentially *ontological* in nature, problems of what kind of interpretations are envisaged. The standard account does not meet these problems. The reason is that it takes its interpretation in the formal set theory of mathematics, and that formal theory already presupposes the ontological commitment of classical logic.

- \top (true) and \perp (false) are formulae in any context.
- Suppose $P \subseteq A_1, \dots, A_n$ is a predicate in the signature, and for each i ($1 \leq i \leq n$), t_i is a term in context \vec{x} of sort A_i . Then $P(t_1, \dots, t_n)$ is a formula in context \vec{x} .
- If s and t are terms in context \vec{x} , and their sorts $\sigma(s)$ and $\sigma(t)$ are the same, then $s = t$ is a formula in context \vec{x} . (We make equality an explicit part of the logic, rather than relying on its introduction as a predicate in the signature.)
- If ϕ and ψ are formulae in context \vec{x} , then so too are $\phi \wedge \psi$ (ϕ and ψ), $\phi \vee \psi$ (*or*), $\phi \rightarrow \psi$ (*implies*) and others to taste.
- If ϕ is a formula in context \vec{x} , then so is $\neg\phi$.
- If ϕ is a formula in context $\vec{x}y$, then $(\exists y)\phi$ and $(\forall y)\phi$ are formulae in context \vec{x} .

Note that it is the *free* variables of a formula that appear in its context – the bound variables do not. However, it is possible for a context to include variables that are not used in the formula. For example, \top and \perp have no free variables but can be considered in any context.

Just as with terms, the interpretation $\{M|\vec{x}.\phi\}$ of a formula in context depends on the context, not just the formula. $\{M|\vec{x}.\phi\}$ will be a subset of $\{M|\sigma(\vec{x})\} = \prod_{i=1}^n \{M|\sigma(x_i)\}$. This allows us to discuss the interpretation of formulae in a more discerning way than if we just took the free variables of a formula to be its context (which is what the standard account in effect does).

The interpretation of formulae in context can now be defined from their structure. Here are the rules. We take $\vec{a} \in \{M|\sigma(\vec{x})\}$, in other words \vec{a} is a list with each a_i in $\{M|\sigma(x_i)\}$.

$\{M|\vec{x}.\top\} = \{M|\sigma(\vec{x})\}$, $\{M|\vec{x}.\perp\} = \emptyset$. (Note how the interpretation depends on the context.)

$\vec{a} \in \{M|\vec{x}.s = t\}$ if $\{M|\vec{x}.s\}(\vec{a}) = \{M|\vec{x}.t\}(\vec{a})$ as elements of $\{M|\sigma(s)\}$.

$\vec{a} \in \{M|\vec{x}.\phi \wedge \psi\}$ if $\vec{a} \in \{M|\vec{x}.\phi\}$ and $\vec{a} \in \{M|\vec{x}.\psi\}$.

$\vec{a} \in \{M|\vec{x}.\phi \vee \psi\}$ if $\vec{a} \in \{M|\vec{x}.\phi\}$ or $\vec{a} \in \{M|\vec{x}.\psi\}$.

$\vec{a} \in \{M|\vec{x}.\neg\phi\}$ if $\vec{a} \notin \{M|\vec{x}.\phi\}$.

$\vec{a} \in \{M|\vec{x}.\phi \rightarrow \psi\}$ if $\vec{a} \notin \{M|\vec{x}.\phi\}$ or $\vec{a} \in \{M|\vec{x}.\psi\}$. (This conforms with the logical equivalence $(\phi \rightarrow \psi) \equiv (\neg\phi \vee \psi)$.)

$\vec{a} \in \{M|\vec{x}.\langle\exists y\rangle\phi\}$ if there is some $b \in \{M|\sigma(y)\}$ such that $\vec{a}b \in \{M|\vec{x}y.\phi\}$.

$\vec{a} \in \{M|\vec{x}.\langle\forall y\rangle\phi\}$ if for every $b \in \{M|\sigma(y)\}$ we have $\vec{a}b \in \{M|\vec{x}y.\phi\}$.

In essence, this is the Tarskian definition of semantics: \wedge is “and”, \vee is “or”, etc.

22.2.1 Interpretations and Ontology

If ontology is the discussion of *being*, or *existence*, then our position is that interpretations are the basis of this discussion. It is the interpretation that provides the instances of formulae. As we have stated it, these instances are elements of *sets*, and at first we understand those as mathematical constructs in formal set theory. However, for any kind of philosophical or applicational discussion we shall want to be able to conceive of M as the “real-world interpretation” of the signature, with each $\{M|\vec{x}.\phi\}$ a collection of real-world things. Though this connection is informal, we shall later look at how this ambition might affect the formal logic.

The use of particular connectives represents an *ontological commitment* that those connectives should have meaning in the setting where we find our interpretations. To see how this could be a problem, let us examine negation. In the example of “there is a lion”, we went to the zoo, saw a lion, and believed. But what about its negation, “there is no lion”? How do we ascertain the truth of that? Certainly it is not enough to visit the zoo and fail to see a lion. Maybe there are lions at the zoo, but they all happen to be asleep in a private part of the cage, or we looked in the sealion pool by mistake. Or maybe there are no lions at the zoo, but there are some on the African savannah. We know how to recognize lions, and we know how to ascertain their existence by seeing one. But that does not tell us at all how to ascertain their non-existence. In other words, there is no uniform ontological account of negation.

Implication is even worse than negation, since negation is a special case of it – $\neg\phi$ is equivalent to $\phi \rightarrow \perp$.

Similarly, there is no uniform ontological account of universal quantification. We might know how to recognize brownness in lions, but that would not tell us how to ascertain the truth of “all lions are brown”.

We shall admit only those formulae that use the connectives to which we are prepared to make the ontological commitment in the interpretations we are considering. For those connectives, we shall take it that the rules given above for determining $\{M|\vec{x}.\phi\}$ still make sense, so $\{M|\vec{x}.\phi\}$ is well defined as a “set” in whatever interpretational sense it is that we have in mind.

22.2.2 Theories and Models

If Σ is a signature, it is usual to define a *theory* over Σ to be a set of *sentences* over Σ , where a sentence is a formula in the empty context. However, we have now envisaged making an ontological restriction to the admissible formulae, and that may rule out implication and negation. It is hardly possible to conduct logic without them, since they lie at the heart of the notion of logical deduction. We shall give a slightly different definition of “theory” that allows for this. This *sequent* form of logic is well known. We shall follow closely the presentation in Johnstone (2002b).

Definition 22.1 A sequent over Σ is an expression $\phi \vdash^{\vec{x}} \psi$ where ϕ and ψ are formulae (with whatever connectives we are using) in context \vec{x} .

This can be read as meaning the sentence $(\forall x_1 \cdots \forall x_n)(\phi \rightarrow \psi)$, but in logics without \rightarrow and \forall this will not be a formula.

Definition 22.2 A theory over Σ is a set T of sequents over Σ , called the axioms of T . An interpretation M satisfies the sequent $\phi \vdash^{\vec{x}} \psi$ if $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\psi\}$, and it is a model of a theory T if it satisfies every axiom in T .

As part of the logic, we shall need to say not only what are the admissible connectives but also what are the *rules of inference*. Each will be presented in the form of a schema

$$\frac{\alpha_1 \cdots \alpha_n}{\beta}$$

where each α_i (a *premiss*) and β (the *conclusion*) is a sequent. We shall not list rules yet, but typical would be the *cut* rule

$$\frac{\phi \vdash^{\vec{x}} \psi \quad \psi \vdash^{\vec{x}} \chi}{\phi \vdash^{\vec{x}} \chi}$$

The *soundness* of a rule is then that if an interpretation satisfies all the premisses it must also satisfy the conclusion. This would normally have to be justified in terms of the ontological explanation of the connectives. For the cut rule it would usually be plain that if $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\psi\}$ and $\{M|\vec{x}.\psi\} \subseteq \{M|\vec{x}.\chi\}$ then $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\chi\}$.

Using the rules of inference, one can infer, or *derive*, many more sequents from the axioms of a theory. If the rules are all sound, then a model of a theory also satisfies all the sequents derived from the axioms.

Note that the sequent formulation (and in particular the explicit context on the turnstile) makes it easier to deal correctly with empty carriers. As an example, consider the two valid entailments $(\forall y)\phi \vdash^x \phi[x/y]$ and $\phi[x/y] \vdash^x (\exists y)\phi$. Applying the cut rule to these we obtain $(\forall y)\phi \vdash^x (\exists y)\phi$. Even if $\sigma(x)$ has an empty carrier this is valid, since then $\{M|x\}$ is the empty set and so are both $\{M|x.(\forall y)\phi\}$ and $\{M|x.(\exists y)\phi\}$. However, the rules do not allow us to deduce $(\forall y)\phi \vdash (\exists y)\phi$ (with empty context), and it would not be valid with the empty carrier because we would have $\{M|\varepsilon.(\forall y)\phi\} = 1$ but $\{M|\varepsilon.(\exists y)\phi\} = \emptyset$.

Unlike the case with formulae, with axioms we make no ontological commitment to being able to ascertain that an interpretation satisfies even a single sequent, let alone a possibly infinite set of them in a theory. There is thus a definite ontological distinction between formulae and sequents. We should understand theories as being like *scientific hypotheses* or *background assumptions*. In fact there is a Popperian flavour to theories.

Suppose we have a theory T and an interpretation M . Suppose also we find some elements in M and ascertain for them some properties from the signature of T . This amounts to finding an element of $\{M|\vec{x}.\phi\}$ for some formula in context $(\vec{x}.\phi)$. The ontological commitment is that we know what is required for our claim to have found such elements. Now suppose also that from the axioms of T we can, using the inference rules, logically deduce the sequent $\phi \vdash^{\vec{x}} \perp$. It should follow that $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\perp\}$. But this is nonsense, since $\{M|\vec{x}.\perp\}$ is by definition empty but we have found an element of $\{M|\vec{x}.\phi\}$. Thus the interpretation M *cannot possibly be a model of T* . If M is a “real world” interpretation, then we cannot simply reject it. Possibly we made a mistake in the way we interpreted ϕ . But if not, then we were mistaken in thinking the axioms of T would apply in the real world. Our observations have led to a Popperian Big No to our theory T .

Note that this process can be carried through only if a sequent $\phi \vdash^{\vec{x}} \perp$ can be derived from the theory T , with ϕ not logically equivalent to \perp . In other words, T must be *falsifiable*. In the example of geometric logic, to which we turn next, this can happen only if T has explicit axioms of the form $\phi \vdash^{\vec{x}} \perp$.

22.3 Geometric Logic

We now turn to *geometric logic*, a positive logic that rejects negation (and also implication and universal quantification) in its formulae. Its ontological commitment is to conjunction, disjunction, equality and existential quantification. Note that we are not claiming it as *the* absolute irreducible logic. We just say that, because of problems with negation and universal quantification, geometric logic is more likely to be applicable in “real world interpretations”. It still carries ontological commitments of its own. For example, consider conjunction. We said that, if ϕ and ψ are formulae in context \vec{x} , then to ascertain that \vec{a} is in $\phi \wedge \psi$ we have to ascertain that \vec{a} is in ϕ and \vec{a} is in ψ . This makes assumptions about our ability to form tuples of things. The logic also presupposes that the two tasks, ascertaining that \vec{a} is in ϕ and ascertaining that \vec{a} is in ψ , do not interfere with each other and can be done in either order. Also, our use of equality means that we expect to be able to ascertain equality between things, but not necessarily inequality. This says something about the kind of things we are prepared to talk about.

The connectives of geometric logic are $\top, \perp, \wedge, \vee, =$ and \exists . However, note one peculiarity: we allow *infinitary* disjunctions \bigvee . If S is a set of formulae, then $\bigvee S$ is also a formula, the disjunction of all the elements of S , and $\{M|\vec{x}.\bigvee S\}$ is defined in the obvious way. This does lead to subtle ontological questions of its own, since

we should examine the nature of the set S and how its members are found. Is S also intended to be a “real-world” collection? If it is purely a mathematical construct, what mathematics are we using? Once we start thinking about different logics, it raises the question of what logic to use for mathematics itself. We shall largely ignore these questions here, except (Section 22.5) to say some brief hints about a fascinating connection between the infinitary disjunctions and algebra.

We shall first present the formal logic, still following Johnstone (2002b), and then (Section 22.3.4) discuss its ontology.

22.3.1 Rules of Inference

The rules of inference for geometric logic as given here are taken from Johnstone (2002b). The first group are *propositional*, in the sense that they have no essential interaction with the terms or variables. The propositional rules are *identity*

$$\phi \vdash^{\bar{x}} \phi,$$

cut

$$\frac{\phi \vdash^{\bar{x}} \psi \quad \psi \vdash^{\bar{x}} \chi}{\phi \vdash^{\bar{x}} \chi},$$

the *conjunction* rules

$$\phi \vdash^{\bar{x}} \top, \quad \phi \wedge \psi \vdash^{\bar{x}} \phi, \quad \phi \wedge \psi \vdash^{\bar{x}} \psi, \quad \frac{\phi \vdash^{\bar{x}} \psi \quad \phi \vdash^{\bar{x}} \chi}{\phi \vdash^{\bar{x}} \psi \wedge \chi},$$

the *disjunction* rules

$$\phi \vdash^{\bar{x}} \bigvee S \quad (\phi \in S), \quad \frac{\phi \vdash^{\bar{x}} \psi \quad (\text{all } \phi \in S)}{\bigvee S \vdash^{\bar{x}} \psi}$$

and *frame distributivity*

$$\phi \wedge \bigvee S \vdash^{\bar{x}} \bigvee \{\phi \wedge \psi \mid \psi \in S\}.$$

Note that $\bigvee \emptyset$ plays the role of \perp (false). To find an element of $\{M|\bar{x}. \bigvee \emptyset\}$ we must find a formula ϕ in \emptyset and then find an element of $\{M|\bar{x}.\phi\}$. But clearly there can be no such ϕ , so $\{M|\bar{x}. \bigvee \emptyset\}$ is empty. From the general disjunction rules we can then derive the rule of *ex falso quodlibet*,

$$\perp \vdash^{\bar{x}} \psi$$

for any ψ in context \bar{x} .

Next come the rules specific to predicate logic. These involve terms and variables.

For the first rule, *substitution*, we use the following notation. Suppose ϕ is a formula in context \vec{x} , and \vec{s} is a vector of terms in another context \vec{y} such that the vector \vec{s} has the same length and sorts as \vec{x} – we can write $\sigma(\vec{s}) = \sigma(\vec{x})$. Then $\phi[\vec{s}/\vec{x}]$ is ϕ with \vec{s} substituted for \vec{x} – the variables in \vec{x} are all replaced by the corresponding terms in \vec{s} . Some notes:

- Since the terms s_i may have their own free variables, taken from \vec{y} , $\phi[\vec{s}/\vec{x}]$ is in the context \vec{y} instead of \vec{x} .
- There is no particular problem if \vec{x} and \vec{y} have variables in common. For example, suppose ϕ is the formula (in context x) $g(x) = a$ and s is the term $f(x)$ where $f : \sigma(x) \rightarrow \sigma(x)$. We can substitute $f(x)$ for x ,

$$(g(x) = a)[f(x)/x] \equiv (g(f(x)) = a).$$

- There can be a problem of “capture of variables” if one of the context variables in \vec{y} is also used as a bound (quantified) variable in ϕ . To avoid this, the bound variables should be renamed to be distinct from the context variables.

The *substitution* rule is

$$\frac{\phi \vdash^{\vec{x}} \psi}{\phi[\vec{s}/\vec{x}] \vdash^{\vec{y}} \psi[\vec{s}/\vec{x}]}$$

The next rules are: the *equality* rules

$$\top \vdash^x x = x, \quad (\vec{x} = \vec{y}) \wedge \phi \vdash^{\vec{z}} \phi[\vec{y}/\vec{x}]$$

In the second \vec{z} has to include all the variables in \vec{x} and \vec{y} , as well as those free in ϕ , and the variables in \vec{x} have to be distinct. Our substitution $\phi[\vec{y}/\vec{x}]$ is not quite in accordance with the definition, since \vec{x} is not the whole of the context. However, we can easily replace it by a licit substitution $\phi[\vec{t}/\vec{z}]$ where \vec{t} is defined as follows. If z_i is x_j for some j , then t_i is defined to be y_j . Otherwise, t_i is defined to be z_i .

The substitution rule justifies *context weakening*

$$\frac{\phi \vdash^{\vec{x}} \psi}{\phi \vdash^{\vec{x},y} \psi}.$$

In other words, a deduction in one context will still be valid if we add extra variables, though not if we remove unused variables (which is what would be done for a deduction of $(\forall x) \phi(x) \vdash (\exists x) \phi(x)$). Note that ϕ here (and ψ likewise) is in two separate contexts: $\vec{x}y$ and \vec{x} . We shall consider it given as in context \vec{x} . Then since \vec{x} can be considered to be a vector of terms in context $\vec{x}y$, we can get ϕ in the extended context as $(\vec{x}y.\phi[\vec{x}/\vec{x}])$.

The *existential* rules are

$$\frac{\phi \vdash^{\vec{x},y} \psi}{(\exists y)\phi \vdash^{\vec{x}} \psi}, \quad \frac{(\exists y)\phi \vdash^{\vec{x}} \psi}{\phi \vdash^{\vec{x},y} \psi}.$$

The *Frobenius* rule is

$$\phi \wedge (\exists y)\psi \vdash^{\vec{x}} (\exists y)(\phi \wedge \psi).$$

22.3.2 Soundness

In a mathematical semantics, the soundness of most of the rules can be readily justified from the semantics of connectives given above. For example, for the final conjunctive rule one has that if $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\psi\}$ and $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\chi\}$ then

$$\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\psi\} \cap \{M|\vec{x}.\chi\} = \{M|\vec{x}.\psi \wedge \chi\}$$

from the definition of $\{M|\vec{x}.\psi \wedge \chi\}$. (In more general semantics we shall see how the rules have subtle consequences for the ontological commitment.)

However, where substitution is involved we have to be more careful. The semantics of a formula is defined in terms of how that formula is structured using the connectives. When a formula is described using a substitution, as in $\phi[\vec{s}/\vec{x}]$, that substitution is not part of the connective structure of the formula and so we do not have a direct *definition* of the “semantics of substitution”. It is nevertheless possible to describe the semantic effect of substitution, but it has to be proved as a *Substitution Lemma*. The Substitution Lemma in effect analyses how substitution interacts with the different connectives.

Recall that each term in context $(\vec{y}.s_i)$ gets interpreted as a function $\{M|\vec{y}.s_i\} : \{M|\sigma(\vec{y})\} \rightarrow \{M|\sigma(s_i)\}$. Putting these together, we get

$$\{M|\vec{y}.\vec{s}\} : \{M|\sigma(\vec{y})\} \rightarrow \prod_i \{M|\sigma(s_i)\} = \{M|\sigma(\vec{s})\} = \{M|\sigma(\vec{x})\}$$

defined by $\{M|\vec{y}.\vec{s}\}(\vec{a}) = (\{M|\vec{y}.s_1\}(\vec{a}), \dots, \{M|\vec{y}.s_n\}(\vec{a}))$.

Lemma 22.3 (Substitution Lemma) *Let \vec{x} and \vec{y} be contexts, and let \vec{s} be a vector of terms in context \vec{y} with $\sigma(\vec{s}) = \sigma(\vec{x})$.*

1. If \vec{t} is a vector of terms in context \vec{x} then $\{M|\vec{y}.\vec{t}[\vec{s}/\vec{x}]\}$ is the composite

$$\{M|\vec{x}.\vec{t}\} \circ \{M|\vec{y}.\vec{s}\} : \{M|\sigma(\vec{y})\} \rightarrow \{M|\sigma(\vec{s})\} = \{M|\sigma(\vec{x})\} \rightarrow \{M|\sigma(\vec{t})\}.$$

2. If ϕ is a formula in context \vec{x} , then $\{M|\vec{y}.\phi[\vec{s}/\vec{x}]\}$ is the inverse image under $\{M|\vec{y}.\vec{s}\}$ of $\{M|\vec{x}.\phi\}$, in other words if $\vec{a} \in \{M|\sigma(\vec{y})\}$ then

$$\vec{a} \in \{M|\vec{y}.\phi[\vec{s}/\vec{x}]\} \text{ iff } \{M|\vec{y}.\vec{s}\}(\vec{a}) \in \{M|\vec{x}.\phi\}.$$

Proof Induction on the structure of t or ϕ . ■

This Lemma is needed for the soundness of the substitution and equality rules. As an illustration of how it is used, consider the substitution rule

$$\frac{\phi \vdash^{\vec{x}} \psi}{\phi[\vec{s}/\vec{x}] \vdash^{\vec{y}} \psi[\vec{s}/\vec{x}]}$$

If $\{M|\vec{x}.\phi\} \subseteq \{M|\vec{x}.\psi\}$ then

$$\begin{aligned} \vec{a} \in \{M|\vec{y}.\phi[\vec{s}/\vec{x}]\} &\Leftrightarrow \{M|\vec{y}.\vec{s}\}(\vec{a}) \in \{M|\vec{y}.\phi\} \\ &\Rightarrow \{M|\vec{y}.\vec{s}\}(\vec{a}) \in \{M|\vec{y}.\psi\} \Leftrightarrow \vec{a} \in \{M|\vec{y}.\phi[\vec{s}/\vec{x}]\}. \end{aligned}$$

One of the more interesting rules here is the second equality rule,

$$(\vec{x} = \vec{y}) \wedge \phi \vdash^{\vec{z}} \phi[\vec{y}/\vec{x}].$$

Recall that \vec{x} here is a sequence of distinct variables from the context \vec{z} , and \vec{y} is a sequence of variables from \vec{z} , not necessarily distinct, that is sort-compatible with \vec{x} . Actually, we might as well assume that \vec{x} is the whole of \vec{z} , since by reflexivity we can add extra equations, for the variables of \vec{z} that are not in \vec{x} , to say that they are equal to themselves. We are therefore justifying $(\vec{x} = \vec{y}) \wedge \phi \vdash^{\vec{z}} \phi[\vec{y}/\vec{x}]$ where each y_i is a variable $x_{\alpha(i)}$, say. Now an element $\vec{a} \in \{M|\vec{x}.\vec{x} = \vec{y}\} \wedge \phi$ is an element $\vec{a} \in \{M|\vec{x}.\phi\}$ such that for each possible index i of the sequence \vec{x} , we have $a_i = a_{\alpha(i)}$. Now consider $\{M|\vec{x}.\phi[\vec{y}/\vec{x}]\}$. Since each y_i is a term in context \vec{x} we have a substitution function $\{M|\vec{x}.\vec{y}\} : \{M|\sigma(\vec{x})\} \rightarrow \{M|\sigma(\vec{x})\}$ mapping \vec{b} to \vec{c} , defined by $c_i = b_{\alpha(i)}$. By the Substitution Lemma, we have $\vec{b} \in \{M|\vec{x}.\phi[\vec{y}/\vec{x}]\}$ if $\vec{c} \in \{M|\vec{x}.\phi\}$. Now given our \vec{a} as above, we can take $\vec{b} = \vec{c} = \vec{a}$ and the required conditions are satisfied. Hence $\vec{a} \in \{M|\vec{x}.\phi[\vec{y}/\vec{x}]\}$.

22.3.3 Beyond Rules of Inference

Each inference rule operates within a single signature, and this imposes a limit on what can be expressed with them. There are more subtle intensions regarding the way different signatures relate to each other. Our main example of this for the moment is the property for mathematical sets that a function is equivalent to a total, single-valued relation – its graph. To express this in logical terms, suppose $(\vec{x}y.\Gamma)$ is a formula in context that is total and single-valued. In other words, it satisfies the properties

$$\begin{aligned} \top \vdash^{\vec{x}} (\exists y)\Gamma \\ \Gamma \wedge \Gamma[y'/y] \vdash^{\vec{x}yy'} y = y'. \end{aligned}$$

Then in any model there is a unique function $f : \sigma(\vec{x}) \rightarrow \sigma(y)$ such that Γ holds iff $y = f(\vec{x})$. This principle is not a consequence of the rules of geometric logic.

Indeed, there are mathematical systems (geometric categories (Johnstone, 2002a) that happen not to be toposes) in which the rules are all sound, but the principle does not hold. Nonetheless, the principle does hold in those systems (toposes) in which geometric logic was first identified, and we take it to be an implicit part of geometric logic. In other words, geometric logic is not just logic (connectives and inference rules). We shall not try to give a complete account of these non-logical principles, though we shall meet some more later.

These principles carry their own ontological commitments. In the above example, the interpretation of a function symbol must be the same as that of a total, single-valued predicate.

22.3.4 Geometric Ontology

We now examine, as carefully as we can, the ontological commitments implicit in geometric logic.

The ontological commitment of the connectives as such does not seem deep. Their interpretation as given above is more or less that of Tarski: \wedge is “and”, \vee is “or”, \exists is “there exists”, etc. But note that the logic does expect something of the “sets” used as carriers. Clearly we must know something about how to find elements of them – how to *apprehend* elements, to use the word of Vickers (1992). To form cartesian products $\{M|\vec{A}\}$, we must also know how to form tuples of elements. This is perhaps not so obvious as it seems. How do you apprehend a tuple of lions? Is it just a bunch of lions? But that would not allow a tuple with the same lion in more than one component (e.g. (Elsa, Lenny, Elsa, Parsley)), which is certainly allowed by the logic. (Otherwise the equality relation is empty.) So clearly the components of the tuple are more like pointers, “that lion over there, Elsa”. And is it properly understood how the interpretation works with observations made at different times? Next, because $=$ (though not \neq) is built in to the logic, we must know something about how to ascertain equality between a pair of apprehended elements.

Let us suppose – in some interpretation M – we know how to apprehend elements and ascertain equality for each sort. (The discussion is not quite finished yet, because we need to examine what properties these ingredients have. We shall return to it later.) Let us suppose we also know how to form tuples. Equality between tuples will be ascertained componentwise. This will then tell us about the sets $\{M|\vec{A}\}$ for each sort tuple \vec{A} . For a predicate $P \subseteq \vec{A}$, the interpretation $\{M|P\} \subseteq \{M|\vec{A}\}$ must tell us what it takes to ascertain $P(\vec{a})$ for $\vec{a} \in \{M|\vec{A}\}$. This then lifts to formulae in context $(\vec{x}.\phi)$.

Note that there may be different ways of ascertaining $\phi(\vec{a})$ for the same \vec{a} , hence different manifestations of the same element of $\{M|\vec{x}.\phi\}$. What is important is that equality between them is determined by equality for the underlying \vec{a} . An illuminating example is when ϕ is of the form $(\exists y)\psi$. To ascertain that \vec{a} is in $\{M|\vec{x}.\phi\}$, one must actually apprehend an element $\vec{a}b$ of $\{M|\vec{x}y.\psi\}$. Hence apprehending an element of $\{M|\vec{x}.\phi\}$ is exactly the same as apprehending an element of $\{M|\vec{x}y.\psi\}$.

But ascertaining equality between them is different, since in the former case the y component is ignored.

Now there is a rather fundamental question about the meaning of a sequent $\phi \vdash^{\bar{x}} \psi$. We have already explained it as meaning $\{M|\bar{x}.\phi\} \subseteq \{M|\bar{x}.\psi\}$. But what does this mean in terms of apprehension? Suppose an element \vec{a} is apprehended in $\{M|\bar{x}.\phi\}$. What does it mean to say it is also in $\{M|\bar{x}.\psi\}$? To put it another way, is it possible to apprehend some \vec{b} in $\{M|\bar{x}.\psi\}$ such that \vec{b} and \vec{a} are equal as elements of $\{M|\sigma(\bar{x})\}$? Three possible interpretations spring to mind.

1. “Already done”: Whatever it took to apprehend \vec{a} as an element of $\{M|\bar{x}.\phi\}$, that is already enough to apprehend a suitable \vec{b} .
2. “Nearly done”: A well defined program of extra work will yield a suitable \vec{b} given \vec{a} .
3. “Can be done”: There is some suitable \vec{b} , though we don’t necessarily know how to find it.

The “already done” interpretation would be extremely strong, since it means that validity of sequents follows directly from knowing how formulae are interpreted. This is clearly incompatible with the idea mentioned above that theory axioms represent background assumptions, or scientific hypotheses.

The “nearly done” interpretation is less strong, since some ingenuity might be required to find the “well defined program of extra work”. In fact, this interpretation is roughly speaking the standard one for intuitionistic logic. There one thinks of the elements of $\{M|\bar{x}.\phi\}$ as the *proofs* of ϕ . A proof of $(\forall \bar{x})(\phi \rightarrow \psi)$ (and so of the sequent $\phi \vdash^{\bar{x}} \psi$) is an algorithm that takes a tuple \vec{a} and a proof of $\phi(\vec{a})$ (in other words, an element of $\{M|\bar{x}.\phi\}$ for some M) and returns a proof of $\psi(\vec{a})$. Nonetheless, it is hard to see this as compatible with the idea of axioms as scientific hypotheses.

We shall follow the “can be done” interpretation.

Note that this makes the cut rule,

$$\frac{\phi \vdash^{\bar{x}} \psi \quad \psi \vdash^{\bar{x}} \chi}{\phi \vdash^{\bar{x}} \chi},$$

more subtle than it looks. Suppose we believe the sequents $\phi \vdash^{\bar{x}} \psi$ and $\psi \vdash^{\bar{x}} \chi$ for an interpretation M , and we want to justify $\phi \vdash^{\bar{x}} \chi$. Suppose we have \vec{a} in $\{M|\bar{x}.\phi\}$. The first sequent tells us that there is, somewhere out there waiting to be found, a \vec{b} in $\{M|\bar{x}.\psi\}$ equal to \vec{a} as elements of $\{M|\bar{x}.\psi\}$. However, it does not tell us how to find it. The second sequent tells us that when we do find it, we can then believe there is a \vec{c} in $\{M|\bar{x}.\chi\}$ equal to \vec{b} . The cut rule asserts that we do not have to go to the trouble of finding \vec{b} . Our belief that it is there, and one day might be found, is already enough to justify us in believing in \vec{c} . Hence we justify the sequent $\phi \vdash^{\bar{x}} \chi$.

We can put this another way. Our explanation of the “can be done” interpretation of a sequent $\phi \vdash^{\bar{x}} \psi$, was that if *we have* an element of $\{M|\bar{x}.\phi\}$, then *there is* (out there somewhere) an equal element of $\{M|\bar{x}.\psi\}$. The cut rule uses the idea that we can equivalently weaken on the left hand side, and start from *there is* an element of $\{M|\bar{x}.\phi\}$.

For geometric logic, “can be done” governs how we interpret function symbols. Recall that a function $f : \vec{A} \rightarrow B$ is expected to be logically equivalent to its graph, a predicate $\Gamma_f \subseteq \vec{A}, B$ (or, more generally, a formula in context) that is total and single-valued:

$$\begin{aligned} & \top \vdash^{\vec{x}} (\exists y) \Gamma_f(\vec{x}, y) \\ \Gamma_f(\vec{x}, y) \wedge \Gamma_f(\vec{x}, y') & \vdash^{\vec{x}, y, y'} y = y'. \end{aligned}$$

These sequents too are given a “can be done” interpretation. Think of the graph Γ_f as being a *specification* of the function. Given arguments \vec{a} and a candidate result b , Γ_f provides a way for ascertaining whether b is indeed the result for $f(\vec{a})$, but it does not in any way tell us how to find b . (That would in fact be a “nearly done” interpretation.) The totality axiom tells us (or hypothesizes) that there is such a b waiting to be found, and single-valuedness says that any two such b s are equal. It follows that when we talk about a “function” between “real world sets”, we must not in general expect this to be a method or algorithm.

This style of interpretation can actually be internalized in the logic by eliminating function symbols in favour of predicates for their graphs (together with axioms for totality and single-valuedness). Suppose we have a graph predicate Γ_f for each function symbol f , characterized by

$$\Gamma_f(\vec{x}, y) \dashv\vdash^{\vec{x}, y} y = f(\vec{x}).$$

Then we can define a graph formula in context $(\vec{x}\vec{y}. \Gamma_{\vec{f}})$ for each term tuple in context $(\vec{x}.\vec{t})$, where $\sigma(\vec{y}) = \sigma(\vec{t})$. For a single term t , if t is a variable x_i then Γ_t is just the formula $y = x_i$. If t is $f(\vec{s})$, suppose we have defined $\Gamma_{\vec{s}}$ in context $\vec{x}\vec{z}$. Then we can define $\Gamma_{f(\vec{s})}$ in context $\vec{x}y$ as $(\exists \vec{z})(\Gamma_{\vec{s}} \wedge \Gamma_f(\vec{z}, y))$. Once that is done, formulae can be replaced by alternatives without function symbols. For example, $P(\vec{t})$ can be replaced by $(\exists \vec{y})(\Gamma_{\vec{f}} \wedge P(\vec{y}))$.

If we look at the inference rules, we find that some of them are obvious, but some have hidden subtleties. We have already mention the cut rule.

The next interesting ones are the conjunction rules. Examining conjunction itself in more detail, to apprehend an element of $\{M|\vec{x}.\phi \wedge \psi\}$, we must apprehend elements \vec{a} and \vec{b} of $\{M|\vec{x}.\phi\}$ and $\{M|\vec{x}.\psi\}$ and then ascertain that they are equal as elements of $\{M|\sigma(\vec{x})\}$. Now consider the rule

$$\frac{\phi \vdash^{\vec{x}} \psi \quad \phi \vdash^{\vec{x}} \chi}{\phi \vdash^{\vec{x}} \psi \wedge \chi}.$$

For the conclusion, suppose we have apprehended \vec{a} in $\{M|\vec{x}.\phi\}$. The premiss sequents tell us that there are \vec{b} and \vec{c} in $\{M|\vec{x}.\psi\}$ and $\{M|\vec{x}.\chi\}$ such that \vec{b} and \vec{a} are equal in $\{M|\sigma(\vec{x})\}$, and so are \vec{c} and \vec{a} . To deduce that there is an element of $\{M|\vec{x}.\psi \wedge \chi\}$, clearly we need to make assumptions about “ascertaining

equality” – it needs to be symmetric and transitive. In fact the equality rule $\top \vdash^x x = x$ will need it to be reflexive too, so it must be an equivalence relation.

For the rules involving substitution, we need to consider the Substitution Lemma. This is most conveniently understood in terms of the logical style explained above, in which function symbols are replaced by predicates for their graphs. The Substitution Lemma (or at least, part (ii) of it) then says the following. Suppose ϕ is in context \vec{x} , and \vec{s} in context \vec{y} is sort compatible with \vec{x} . Then finding an element $\vec{a} \in \{M|\vec{y}.\phi[\vec{s}/\vec{x}]\}$ is equivalent (in the “can be done” sense) to finding elements $\vec{a}\vec{b} \in \{M|\vec{y}\vec{x}.\Gamma_{\vec{s}}\}$ and $\vec{c} \in \{M|\vec{x}.\phi\}$ such that \vec{b} is equal to \vec{c} .

22.4 Topology

The links between geometric logic and topology arise from a very direct correspondence: the disjunctions and finite conjunctions in the logic correspond to the unions and finite intersection that characterize the behaviour of open sets. There is a then a rough correspondence between propositional geometric theories and topological spaces: the space is the space of models for the theory, topologized using the logical formulae.

Using the theories instead of topological spaces is generally known as “point-free topology”, and has been found useful in various fields, especially in constructive mathematics (e.g. as “locales” (Johnstone, 1982), in topos theory, and as “formal topologies” (Sambin, 1987) in predicative type theory). The applications in computer science, based on ideas of observational theory, could even be read as suggesting that topology in some sense arises from an ontological shift in the understanding of propositions.

A major idea in topos theory is to generalize this correspondence to predicate theories, leading to Grothendieck’s new notion of *topos* as “generalized topological space”. The theory then corresponds to its “classifying topos”, representing (in an indirect way) the “space of models”. These ideas are implicit in the standard texts on toposes, such as MacLane and Moerdijk (1992), Johnstone (2002a, b), though often hidden. Vickers (2007) attempts to bring them out more explicitly.

In the space available here it has only been possible to hint at the deep connections between geometric logic and topology, but the curious reader is encouraged to explore the references suggested.

22.5 Algebra

We now turn to a feature of geometric logic that makes essential use of the infinitary disjunctions, and sets it quite apart from finitary logics. The effect is that geometric logic can be considered to embrace a variety of set-theoretic constructions on sorts, and we shall examine the ontological aspects of this.

22.5.1 Lists and Finite Sets

In any geometric theory T , suppose A is a sort. Consider now an extended theory that also has a sort B , together with function symbols

$$\begin{aligned}\varepsilon &: B \\ \gamma &: A \times B \rightarrow B.\end{aligned}$$

We shall in fact use infix notation for γ , writing $x \circ y$ for $\gamma(x, y)$. We also add axioms

$$\begin{aligned}x \circ y &= \varepsilon \vdash^{xy} \perp \\ x \circ y &= x' \circ y' \vdash^{xyx'y'} x = x' \wedge y = y' \\ \top &\vdash^y \bigvee_{n \in \mathbb{N}} (\exists x_1) \cdots (\exists x_n) (y = x_1 \circ \dots \circ x_n \circ \varepsilon).\end{aligned}$$

Here \mathbb{N} denotes the set $\{0, 1, 2, 3, \dots\}$ of natural numbers, so the right hand side of the last axiom is

$$y = \varepsilon \vee (\exists x_1) y = x_1 \circ \varepsilon \vee (\exists x_1)(\exists x_2) y = x_1 \circ (x_2 \circ \varepsilon) \vee \dots$$

In any model M of this extended theory, each list (a_1, \dots, a_n) of elements of $\{M|A\}$ gives an element $a_1 \circ \dots \circ a_n \circ \varepsilon$ of $\{M|B\}$. The third axiom says that *any* element of $\{M|B\}$ can be got this way, and the first two axioms say that the list is unique – if

$$a_1 \circ \dots \circ a_m \circ \varepsilon = a'_1 \circ \dots \circ a'_n \circ \varepsilon$$

then $m = n$ and each $a_i = a'_i$. It follows that $\{M|B\}$ is isomorphic with the set of finite lists of elements of $\{M|A\}$, which we write $\{M|A\}^*$.

This ability to characterize list sets (up to isomorphism) by logic relies essentially on the infinitary disjunctions in geometric logic. It cannot be done in finitary logic. It means that in effect geometric logic embraces sort constructors. Instead of adding all the axioms explicitly, we could allow ourselves to write a derived sort A^* , with the interpretation $\{M|A^*\} = \{M|A\}^*$.

Moreover, this fits with our previous ontology. To apprehend an element of $\{M|A^*\}$, we should apprehend a tuple \vec{a} of elements of $\{M|A\}$. The tuple can have any finite length. To ascertain that \vec{a} and \vec{a}' are equal, we should find that they have the same length and then that each component of \vec{a} is equal to the corresponding component of \vec{a}' .

In a similar way, we can use geometric logic to characterize the *finite power set*, $\mathcal{F}A$. We use the same symbols ε and \circ , but now ε is to mean the empty set \emptyset and $a \circ b$ means $\{a\} \cup b$. Hence $a_1 \circ \dots \circ a_n \circ \varepsilon$ means $\{a_1, \dots, a_n\}$. We keep the third axiom, but we replace the first two so as to give a different definition of equality. For this we take axioms

$$x_1 \circ \dots \circ x_m \circ \varepsilon = x'_1 \circ \dots \circ x'_n \circ \varepsilon \dashv\vdash^{\vec{x}'} \bigwedge_{i=1}^m \bigvee_{j=1}^n x_i = x'_j \wedge \bigwedge_{j=1}^n \bigvee_{i=1}^m x'_j = x_i$$

for all possible m, n . This in effect says $\{a_1, \dots, a_m\} = \{a'_1, \dots, a'_n\}$ iff $\{a_1, \dots, a_m\} \subseteq \{a'_1, \dots, a'_n\}$ (i.e. every a_i is equal to at least one of the a'_j s) and $\{a'_1, \dots, a'_n\} \subseteq \{a_1, \dots, a_m\}$.

Again, this fits our ontology. To apprehend an element of $\{M|\mathcal{F}A\}$, we should apprehend a tuple \vec{a} of elements of $\{M|A\}$, just as we did for $\{M|A^*\}$. However, this time the equality is different.

22.5.2 Free Algebras

The list sets and finite power sets are both examples of a much more general construction, of *free algebras*. These arise from a particular kind of geometric theory, namely *algebraic* theories. An algebraic theory is defined by *operators* and *equational laws*, and in terms of geometric theories as defined above this means there are no predicates, and the axioms are all of the form

$$\top \vdash^{\vec{x}} s = t.$$

The models are then often called *algebras*.

Many examples are widely known, for example the theories of groups, rings, vector spaces and Boolean algebras.

The fact that algebraic theories are geometric is interesting, but not very deep. A much more significant fact about geometric theories emerges when one considers *free algebras*, and this is something that relies on very specific properties of geometric logic, and in particular its use of infinitary disjunctions.

Let T be an algebraic theory, with only one sort, A . (Similar results hold for theories with more than one sort, but they are more complicated to state.) A *free algebra*, on a set X , is constructed in two stages. First, we consider all the terms that can be formed, in the empty context, using the operators of T , and also using the elements of X as constants. Next, we define two terms s and t to be *congruent* if the sequent $\top \vdash s = t$ can be inferred (using the inference rules of geometric logic) from the axioms of T . The set of congruence classes is an algebra for T , and is called the *free T -algebra* on X , denoted $T\langle X \rangle$. It can be proved to have a characteristic property that is actually rather fundamental: given any T -algebra A , then any function $f : X \rightarrow A$ extends uniquely to a T -homomorphism from $T\langle X \rangle$, got by evaluating the terms (representing elements of $T\langle X \rangle$) in A .

List sets and finite powersets are both examples of free algebras.

The *logical* significance of these constructions is that in geometric theories, geometric structure and axioms can be used to constrain the carrier for one sort to be isomorphic to a free algebra over the carrier of another. (This is not possible in *finitary* first order predicate logic.) Hence geometric logic may be understood as

including an inherent *type theory*, with constructions that can be applied to sorts. The list sets and finite powersets were the first examples. This again is something that goes beyond the strict logic, analogous to issues discussed in Section 22.3.3.

On the other hand, there is also an *ontological* significance. Construction of terms can be understood as a process of apprehending elements (by gathering together other elements in a structured way), and then finding a proof of congruence is ascertaining equality between elements. Thus we may see a “real-world” significance in the free algebra constructions, as typified by list sets and finite powersets.

22.6 Categories

We have described a connection between, on the one hand, the formal structure of formulae, constructed using formal symbols such as \wedge and \vee , and, on the other, the informal ideas of how we might interpret those formulae in the real world. At first sight the interpretation is straightforward, once we have assigned meaning to the primitive symbols of the signature. After that one might think it is just a matter of interpreting \wedge as “and”, \vee as “or”, and so on. However, we saw that particular connectives could easily be problematic. Having particular connectives and particular logical rules about their use imposes an ontological commitment on our interpretations.

This comparison between the logic and the real world may seem unavoidably vague, because of the transformation from formal to informal. However, it actually has two separate transformations bundled up together: one is from formal to informal, but the other is from a logical formalism of terms and formulae to an explanation that is more about collections and functions. There is a way to separate these out using *category theory*. A category is a mathematical structure whose “objects” and “morphisms” may embody intuitive ideas of collections and functions between them. The idea is discussed in other chapters in this collection, and in particular that of Johnson and Rosebrugh. It is that the formal category structure can represent – and more directly than logic – the informal structure of the real world that we want to capture.

For example, one of the assumptions we made, in Section 22.3.4, of real world objects was that it is possible to yoke them together in pairs or longer tuples. This corresponds directly to the categorical idea of *product*. If X and Y are two collections, then there should be another collection $X \times Y$ whose elements are pairs of elements, one from X and one from Y . More generally, if we have two functions $f : Z \rightarrow X$ and $g : Z \rightarrow Y$, then we should be able to pair their results to get a function $\langle f, g \rangle : Z \rightarrow X \times Y$. Category theory uses this idea to characterize $X \times Y$ as the “product” of X and Y , so the informal idea of pairing elements corresponds naturally to the formal idea that the category has products.

In the formal world, on the other hand, we can transform from the logical style to the collections style, by interpreting the logic inside a category. This is known as *categorical logic*. The ontological commitment can now be discussed in a precise mathematical way in terms of the properties of the category. Then each logical

theory whose connectives and rules respect that ontological commitment gives rise to a *theory category* (see Barrand Wells (1984); also known as *classifying category*) with those properties, freely constructed so as to have in it a model of the logical theory. The relation between theory and category is in fact similar to the way a *sketch* presents a category, as discussed in [Chapter 24](#), by Johnson and Rosebrugh, this volume.

More on how geometric logic corresponds to categorical structure, with pointers to reading the standard texts such as Johnstone (2002b) and MacLane and Moerdijk (1992), can be found in Vickers (2007).

22.6.1 Sheaves

Sheaves provide a fundamental example of a formal setting where geometric logic can be interpreted, but other parts of ordinary logic – including negation – go wrong. They cannot support the ontological commitment of full classical logic.

We shall not define sheaves in detail here. A good intuition is that if we have a topological space X , then a sheaf over it is a set “continuously parametrized” by a point of x . For each x there is a set A_x (the *stalk* of the sheaf at x), and as x varies, the stalk, the set A_x , varies with it in a continuous way – no sudden jumps. If $a \in A_x$ then there is a neighbourhood U of x such that for each $y \in U$, the stalk A_y has an element corresponding to a . Also, if there are two such ways of choosing “elements corresponding to a ”, then there is some neighbourhood of x where the two choices agree. That is very vague, but it can be made precise and defines the notion of “local homeomorphism” (see, e.g. Vickers (2007) again).

Without saying any more about the general notion, we can describe a very simple example where the problems with negation are easy to see. *Sierpinski space* has two points, \perp and \top . The topology can be described using the idea of *neighbourhoods*, referred to above. $\{\top\}$ is a neighbourhood of \top , but the only neighbourhood of \perp is $\{\perp, \top\}$. When one works out what a sheaf is, it turns out to be a pair of sets A_\perp and A_\top (the stalks), together with a function $f : A_\perp \rightarrow A_\top$. The function is needed because, for each $a \in A_\perp$, the definition of sheaf requires a neighbourhood U of \perp (and in this case U can only be $\{\perp, \top\}$), and an element $a_\top \in A_\top$ corresponding to a . The function f shows how to pick a_\top for each a .

Subsheaves are analogous to subsets. A subsheaf of the sheaf A (given by $f : A_\perp \rightarrow A_\top$) is a pair of subsets $B_\perp \subseteq A_\perp$ and $B_\top \subseteq A_\top$ such that when f is restricted to B_\perp , it maps into B_\top :

$$\begin{array}{ccc} B_\perp & \subseteq & A_\perp \\ \downarrow & & \downarrow f \\ B_\top & \subseteq & A_\top \end{array}$$

Now suppose we have another subsheaf, C . We can try to define more subsheaves $B \cup C$ and $B \cap C$ “stalkwise” by

$$\begin{aligned} (B \cup C)_x &= B_x \cup C_x, \\ (B \cap C)_x &= B_x \cap C_x. \end{aligned}$$

for $x = \perp, \top$. But are these subsheaves? The question is whether f restricts properly. In fact, for \cup and \cap it works. This shows that the geometric connectives \vee and \wedge can be interpreted in the expected way.

Now let us look at \neg . We try to define a subsheaf $\neg B$ by $(\neg B)_x = A_x - B_x = \{a \in A_x \mid a \notin B_x\}$. Our question about f now amounts to the following. We know that if $a \in B_\perp$ then $f(a) \in B_\top$. Can we deduce that if $a \notin B_\perp$ then $f(a) \notin B_\top$? No, in general. For a simple example, take $B_\perp = \emptyset, B_\top = A_\top$.

For an intuitive idea of what is happening here, think of A_\top as “the reality of A ”, and A_\perp as “what we have seen of it”. f translates our observations into real things. However, (i) we may not have seen everything – there may be elements of A_\top that are not f of anything; and (ii) we may have observed two things that are in reality one and the same. Now \vee and \wedge work just as well for our observations as for reality, but \neg doesn’t. Failure to observe (calculating $A_\perp - B_\perp$) does not map to non-existence ($A_\top - B_\top$).

References

- Barr, M., and C. Wells. 1984. *Toposes, triples and theories*. New York, NY: Springer, reissued as Barr and Wells, 2005.
- Barr, M., and C. Wells. 2005. *Toposes, triples and theories*. Reprints in *Theory and Applications of Categories*, no. 12. *Theory and Applications of Categories*, Mount Allison University, originally published as Barr and Wells, 1984.
- Johnstone, P.T. 1982. *Stone spaces*, Cambridge Studies in Advanced Mathematics, no. 3, Cambridge, UK: Cambridge University Press.
- Johnstone, P.T. 2002a. *Sketches of an elephant: A topos theory compendium*, vol. 1, Oxford Logic Guides, no.44, Oxford: Oxford University Press.
- Johnstone, P.T. 2002b. *Sketches of an elephant: A topos theory compendium*, vol. 2, Oxford Logic Guides, no. 44, Oxford: Oxford University Press.
- Mac Lane, S., and I. Moerdijk. 1992. *Sheaves in geometry and logic*. New York, NY: Springer.
- Russell, B. 1945. *History of western philosophy*. New York, NY: Simon and Schuster.
- Sambin, G. 1987. Intuitionistic formal spaces – a first communication. In *Mathematical logic and its applications*, ed. D.G. Skordev, 187–204. Cambridge: Plenum.
- Vickers, S. 1989. *Topology via logic*. New York, NY: Cambridge University Press.
- Vickers, S.J. 1992. Geometric theories and databases. In *Applications of categories in computer science LMS Lecture Note Series*, 177, eds. M.P. Fourman, P.T. Johnstone, and A.M. Pitts, 288–314. Cambridge: Cambridge University Press.
- S. Vickers. 2007. Locales and toposes as spaces. In *Handbook of spatial logics*, eds. M. Aiello, I.E. Pratt-Hartmann, and J.F.A.K. van Benthem, 429–496. New York, NY: Springer.

Chapter 23

The Institutional Approach

Robert E. Kent

Systems, scientific and philosophic, come and go. Each method of limited understanding is at length exhausted. In its prime each system is a triumphant success: in its decay it is an obstructive nuisance.

Alfred North Whitehead, *Adventures of Ideas*

23.1 Introduction

The institutional approach for the logical semantics of ontologies provides a principled framework for their modular design; in particular, it provides a natural framework for formulating the “lattice of theories” (LOT) approach to ontological organization. According to Sowa, the purpose of LOT is to provide a systematic way of relating all possible ontologies in order to facilitate their inevitable upgrades and conversions. The goal of LOT is to create a framework “which can support an open-ended number of theories (potentially infinite) organized in a lattice together with systematic metalevel techniques for moving from one to another, for testing their adequacy for any given problem, and for mixing, matching, combining, and transforming them to whatever form is appropriate for whatever problem anyone is trying to solve” (Sowa, 2000). The theories of Institutions (Goguen and Burstall, 1992), Information Flow (Barwise and Seligmann, 1997) and Formal Concept Analysis (Ganter and Wille, 1999) have independently formulated and developed various concepts surrounding the LOT construction. But the institutional approach gives the most comprehensive treatment.

Within an institution the lattice of theories is the indexing of the context of theories by the context of languages (an index can be thought of as a list or a pointing device; a context (Goguen, 1991) represents a kind of mathematical structure, such as algebraic structure, topological structure or logical structure; contexts and indexed contexts are discussed below). The central relation inside the lattice of

R.E. Kent (✉)
Ontologos, Pullman, Washington, USA
e-mail: reKent@ontologos.org
(in memory of Joseph Goguen)

theories is logical entailment. However, in problem solving we are always supposed to be willing to “think outside the box.” In this situation, the institutional approach instructs us to think outside the lattice of theories. Inside is the entailment relation between theories, but outside are links between theories. Theory links specialize to theory entailment within the fiber over a language (this fiber consists of all theories having that language; the idea of a fiber is defined in the section on passages). Theory links (discussed further below) are logical language maps between theories that preserve entailment. Theories and links between theories form the context of theories.

The institutional approach starts with the motivation to unify the numerous efforts to use logic for the representation and organization of the knowledge space of various communities. In order to accomplish this, the institutional approach uses the two related distinctions (the general versus the special) and (the abstract versus the concrete). Both generalization and abstraction can have many levels or degrees. The correct level or degree depends on the goal in mind. In order to reach a certain level of generality we need to abstract from the unimportant and superfluous details, but still retain the essential ones. In the institutional approach, a logical system is identified with an institution. The institutional approach, whose goal is the representation and maintenance of systems of ontologies, generalizes by abstraction over various logical systems such as first order, second order, higher order, Horn clause, equational, temporal, modal and infinitary logics.

This chapter discusses the institutional approach within the theory and application of ontologies. One caveat: although the institutional approach to ontologies extensively uses category theory, this chapter has not been written for a reader with background knowledge of category theory. Instead this chapter has been written for philosophers, computer scientists and the general public. For this reason less technical and more common terminology has been used in describing the basic concepts. Such an approach has been used before. Goguen has used the term “(mathematical) context” for the category concept (this may be an especially useful alternative for philosophers), Manes has used the term “passage” for the functor concept, and Lawvere and others regard the concept of adjoint functors as a generalized “inverse pair” of functors. There is a key¹ for this terminology. In addition, very few abstract

¹Here is a key to the terminology used in this paper.

This Paper	Category theory
Object	Object
Link	Arrow, morphism, 1-cell
Connection	2-cell
Beginning, origin(ation)	Source, domain
Ending, destination	Target, codomain
Context	Category
Passage	Functor
Bridge	Natural transformation
Invertible passage	Adjunction
Equivalence	Natural equivalence
(Co)relation	(Co)cone
Sum (product)	Colimit (limit)
Relative sum	Left Kan extension

symbols have been used. This is a mixed blessing, since, although the intimidation of abstract mathematics has been removed, the advantage of the extremely useful idea of “commutative diagram” (studded with abstract symbols) cannot be easily used. For example, the general discussion about the architecture only uses the symbol **V** for a general ambient context and the symbol **CIs** for the specific ambient context of classifications (an ambient context is a background encompassing context within which we form diagrams; the institutional approach uses the ambient context **CIs** of classifications and infomorphisms). For an overview of category theory, see [Chapter 21](#), by Healy, this volume.

23.1.1 Ontologies

Ontologies are of two types: populated and unpopulated. Populated ontologies contain instance data, whereas unpopulated ontologies do not. Instances (tokens, particulars) are things that are classified, whereas types (universals) are things that classify. “Aristotle” is a particular individual in the ancient world, whereas “human” and “philosopher” are types that classify and describe that individual. “It is particulars, things in the world, that carry information; the information they carry is in the form of types” (Barwise and Seligman, 1997).

Any ontology is situated within the context of the logical language of a domain (of discourse), which often consists of the generic ideas of the connectives and quantifiers from logic and the specific ideas of the signature (the constant, function and relation symbols) for that context (Goguen and Burstall, 1992). An unpopulated ontology is represented as a theory consisting of a collection of statements or sentences based on the language. The theory allows for the expression of the laws and facts deemed relevant for the domain. A structure of a domain provides a universe of discourse in which to interpret statements of a theory. Both theory and structure are described and constrained by the logical language. A populated ontology is represented as a (local) logic or knowledge base having two components, a theory and a structure that share the same underlying language. This notion of logic is a precursor to the local logics defined and used in Information Flow (Barwise and Seligman, 1997), which are more closely represented by the composite logics defined below. In general, the logics in the institutional approach are neither sound nor complete. A logic is sound when each sentence of the theory is true when interpreted in the structure; that is, when the structure satisfies each sentence of the theory. A logic is complete when every sentence satisfied by the structure is a sentence entailed by the theory. Associated with any structure is a natural logic whose theory consists of all sentences satisfied by the structure. The natural logic is essentially the only sound and complete logic over a given language. Associated with any logic is its restriction – the sound logic with the same underlying structure, whose theory consists of all sentences satisfied by the structure and entailed by the theory. There is a projective component passage from logics to structures. In the opposite direction, there is a natural logic passage from structures to sound logics. With structure projection and natural logic, the context of structures forms a reflective subcontext of the context of sound logics. There is a restriction passage from logics to sound logics. With restriction and inclusion, the context of sound logics forms a coreflective

subcontext of the context of logics. A composite logic, the abstract representation of the (local) logics of the theory of Information Flow (Barwise and Seligman, 1997), consists of a base logic and a sound logic sharing the same language and theory, where any sentence satisfied by the base logic structure is also satisfied by the sound logic structure. Composite logics form a context with two projective component passages to general logics and sound logics. Information systems (see below) can be defined in either theories, logics, sound logics or composite logics. In the sketch institution **Sk**, the category-theoretic approach to ontological specification (discussed in detail below), a sound logic is a sketched interpretation (generalized universal algebra) consisting of a sketch and an interpretation that satisfies that sketch. In the logical environment **IFC**, the basic institution for Information Flow (defined below), a logic consists of a classification (structure) and a theory sharing a common set of types (language); in addition, there is a subset of instances, called normal instances, which satisfy all sequents (sentences) in the theory. The local logics of Information Flow are the composite logics of **IFC** with the classification and theory providing the base logic component and restriction to the normal instances providing the sound logic component.

Institutions, which represent logical systems, abstract and generalize the semantic definition of truth (Goguen and Burstall, 1992), which consists of a relation of satisfaction between structures and sentences. In short, the institutional approach is abstract model theory. The first step of abstraction in the institutional approach is to regard each structure as an instance, each sentence as a type and each occurrence of satisfaction as a classification: a structure satisfies a sentence when the sentence-as-type classifies the structure-as-instance. In this regard, the theory of institutions and the theory of Information Flow are very compatible; indeed, one can regard the theory of institutions as an indexed or parametric theory of information flow, with each institution (the parameter) having a theory of information flow constructed over it and links between institutions (parameter map) providing comparisons between these theories of information flow. The second step of abstraction in the institutional approach is to regard the logical language, which provides the context for an ontology, to be an indexing object. The institutional approach refers to such an indexing object as a language (signature in Goguen and Burstall, 1992). The language of an institution often contains nonlogical symbols for constants, functions and relations. However, in the institutional approach such a detailed description is inessential, and hence is ignored in the abstraction. In summary, each ontology is indexed by a language and described by a classification.

The representational power of the institutional approach comes from the linkages between objects, such as languages and classifications. Languages are linked by language morphisms, which typically map the constant, function and relation symbols of one language to the constant, function and relation symbols of another language. A classification has instance and type collections and a classification relation between the two. For example, cars are classified by the combination structure-make-year – a particular car is an instance, and the combination “Honda-Civic-1987” is a type. Classifications are linked by infomorphisms, which map between instance collections in the reverse direction (the instance map is said

to be contravariant), map between type collections in the forward direction (the type map is said to be covariant), and require invariance of classification: a type classifies at the origin the image of an instance if and only if the image of the type classifies at the destination the instance. Just as languages index classifications in the institutional approach, so also language morphisms index infomorphisms. When two ontologies are indexed by languages linked by a language morphism, and described by two classifications, then the language morphism indexes an infomorphism that links the two classifications, thereby relating the two ontologies by invariance of satisfaction (invariance of truth under change of notation).

23.1.2 Semantic Integration

An information system (Barwise and Seligman, 1997) is a diagram of ontologies, where each ontology is represented as a logic or a theory. Since each logic (theory) has an underlying structure (language), an information system has an underlying distributed system, which is a diagram of structures (or languages). A channel over a distributed system is a corelation that covers the system (satisfies its semantic alignment constraints) with its vertex called the core of the channel. The optimal channel over a distributed system with sum vertex is an optimal (most refined) covering corelation. Semantic integration² (Kent, 2004; Goguen, 2006) is the two-step process of alignment and then closure. The *alignment* of a distributed collection of ontologies is a human-oriented and creative process that builds a suitable information system.

²We describe the semantic integration of ontologies in terms of theories.

Alignment: Informally, identify the theories to be used in the construction. Decide on the semantic interconnection (semantic mapping) between theories. This may involve the introduction of some additional mediating (reference) theories. Formally, create a diagram of theories of shape (indexing) context that indicates this selection and interconnection. This diagram of theories is transient, since it will be used only for this computation. Other diagrams could be used for other sum constructions. Compute the base diagram of languages with the same shape. Form the sum language of this diagram, with language summing corelation. Being the basis for theory sums, language sums are important. They involve the two opposed processes of “summing” and “quotienting”. Summing can be characterized as “keeping things apart” and “preserving distinctness”, whereas quotienting can be characterized as “putting things together”, “identification” and “synonymy”. The “things” involved here are symbolic, and for a first order logic institution may involve relation type symbols, entity type symbols and the concepts that they denote.

Unification: Form the sum theory of the diagram of theories, with theory summing corelation. The summing corelation is a universal corelation that connects the individual theories in the diagram to the sum theory. The sum theory may be virtual. Using direct flow, move the individual theories in the diagram of theories from the “lattice of theory diagrams over the language diagram” along the language morphisms in the language summing corelation to the lattice of theories over the language sum, getting a homogeneous diagram of theories with the same shape, where each theory in this direct flow image diagram has the same sum language (the meaning of homogeneous). Compute the meet of this direct flow diagram within the fiber “lattice of theories” over language sum, getting the sum theory. The language summing corelation is the base of the theory summing corelation. Using inverse flow, move the sum theory from the language sum back along the language morphisms in the language summing corelation to the language diagram, getting the system closure.

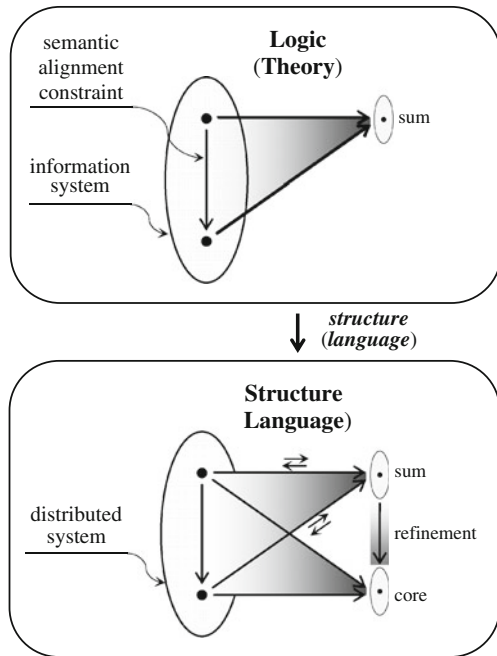
Alignment is called intersection in [Chapter 24](#), by Johnson and Rosebrugh, this volume. The *closure* of the information system is an automatic process of information flow. Closure has three phases: the first two phases are called unification (i) direct flow of the distributed system along the optimal channel (summing correlation over the underlying distributed system), and (ii) meet expansion of the direct flow image within the lattice of logics (theories) indexed by the sum distributed system; the last phase is called projective distribution (iii) inverse flow back along the same optimal channel. Unification is called theory blending in [Chapter 21](#), by Healy, this volume. For further details on the concepts discussed here, see [Chapter 4](#), by Kalfoglou and Schorlemmer, this volume.

The logics (or theories) in the alignment diagram represent the individual ontologies, and the links between logics (or theories) in the alignment diagram represent the semantic alignment constraints. The alignment diagram is an information system with individual logics (or theories) representing parts of the system. The closure of an information system may be relative (partial) or absolute (complete) – relative closure is defined along an indexing passage, whereas absolute closure is defined along the unique indexing passage to the terminal indexing context. Absolute closure can be approximated by indexing passage composition. The relative sum information system along an indexing passage represents the whole system in a partially centralized fashion with the target indexing context defining the degree of centralization. The sum logic in the construction of the absolute closure of an information system represents the whole system in a centralized fashion, whereas the original information system and its closure represent the whole system in a distributed fashion. Ignoring the semantic constraints in the closure information system, the absolute closure is called the distributed logic of the information system in (Barwise and Seligman, 1997) (see also [Chapter 4](#), by Kalfoglou and Schorlemmer, this volume); but it is probably best to recognize it as an information system in its own right and to understand the properties of the closure operator.

Since logics (theories) over the same language are ordered by entailment, information systems with the same indexing context are ordered pointwise by entailment. Two information systems are pointwise entailment ordered when the component logics at each index are entailment ordered. Two information systems are ordered by system entailment when the closure of the first information system is pointwise entailment below the second information system. Hence, the following analogies hold between theories and information systems: theory closure \leftrightarrow system closure; reverse subset order for theories \leftrightarrow pointwise entailment order for systems; and theory entailment \leftrightarrow system entailment. A very important problem in distributed logic is the understanding of how one part of a system affects another part. System closure provides a solution. System closure, which is a closure operator with respect to reverse pointwise entailment order, describes how a distal part (ontology) of the system constrains a proximal part (ontology) of the system.

Figure [23.1](#) provides a graphic representation of semantic integration: in the Logic (Theory) context information systems are represented as ovals, ontologies are represented as nodes within ovals, and alignment constraints between ontologies are represented as edges within ovals; and in the Structure (Language)

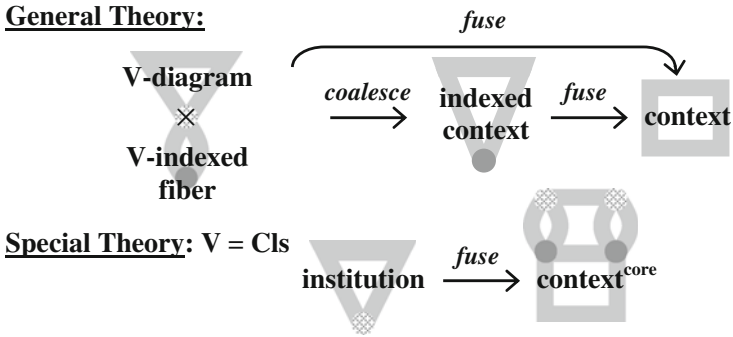
Fig. 23.1 Semantic integration



context distributed systems are represented as ovals, structures (languages) are represented as nodes within ovals, and channels are represented as triangles. The detailed theory of semantic integration in logical environments via system closure is developed in (Kent, 2008, Distributed logic: Information flow in logical environments, unpublished). A simple example of semantic integration, (analogous to one discussed in Chapter 4, by Kalfoglou and Schorlemmer, this volume), starts with a collection of two ontologies represented as theories: (1) the alignment step might create a third bridging, mediating or alignment theory, whose types represent equivalent pairs of types in the two original ontologies, with two mappings from these mediating types back to the types in the equivalent pairs – the alignment diagram (information system) would then have indexing context generated by an inverted vee-shaped graph with three nodes and two edges; (2) the two phases in unification create a sum theory, and the projective distribution phase creates the absolute system closure information system (this corresponds to the integration theory mentioned in Chapter 4, by Kalfoglou and Schorlemmer, this volume). For a more general and heterogeneous (across logical systems) approach to semantic integration, see the paper (Schorlemmer and Kalfoglou, 2008).

23.1.3 Architecture

Here we give a quick overview of the architecture (unifying or coherent form or structure) of the institutional approach; later sections provide more detail. Most of



The **context** icon is a rectangle, which naturally represents a bridge between two passages, with origin on the top side and destination on the bottom. These linked passages have originating context on the left and destination context on the right.



The **indexed context** icon is a large triangle, which represents a link between two indexed contexts, with origin on the left side and destination on the right. These linked indexed contexts have indexing contexts at the top and the context of contexts (the dark circular disk) at the bottom.



The **diagram** icon is a small triangle, which represents a link between two diagrams, with origin on the left side and destination on the right. These linked diagrams have indexing contexts at the top and the ambient context V (the stippled circular disk) at the bottom. Institutions are the special case $V = Cls$.



The **fiber** icon is a lens-shaped figure, which represents a link between two indexed contexts having the same fixed indexing context (the stippled circular disk) at the top, with origin on the left side and destination on the right. The context of contexts (the dark circular disk) is at the bottom.



The combined **diagram-fiber** icon (a product notion) represents a pair of links, a link between diagrams at the top and a link between fibered indexed contexts at the bottom. In a diagram-fiber the ambient context of the diagrams (the stippled circular disk) is the indexing context of the fibered indexed contexts.



The combined **fiber-context** icon (an exponential notion) represents a link between two fiber-to-context passages, with origin on the left side and destination on the right. These linked fiber-to-context passages map a core diagram of fibered indexed contexts at the top to a context at the bottom.

Fig. 23.2 Architecture

the sections in this chapter discuss elements illustrated in Fig. 23.2. Where these and related elements are defined in this chapter, they are italicized. The architecture of the institutional approach to the theory and application of ontologies is illustrated in this figure. The first thing that we see is the separation of the architecture into a general and a special theory. Dual notions (reversed linkage orientation) exist in both

general and special theories. The three basic ideas in the general theory are contexts (rectangle), indexed contexts (large triangle) and their indexed fibers (lens-shaped figure) and diagrams (small triangle). The two basic processes (passages) in the general theory are coalescence and fusion. Coalescence is composition. For an ambient context \mathbf{V} , coalescence operates on \mathbf{V} -diagrams and \mathbf{V} -indexed contexts and returns index contexts (these terms are defined below). Here \mathbf{V} is a valuation context for diagrams and an indexing context for indexed contexts; coalescence bonds diagrams to indexed contexts. In the special case where \mathbf{V} is the context classification \mathbf{Cls} (\mathbf{Cls} -diagrams are institutions), by holding the institution (\mathbf{Cls} -diagram) fixed and letting S denote its indexing language context, we can regard coalescence as a map from the core context of \mathbf{Cls} -indexed contexts to the core context of S -indexed contexts; for example, mapping the logic indexed context for classifications to the logic indexed context for the fixed institution (by applying fusion to this process, in Fig. 23.8 we define a logic map from the diamond-shaped core diagram for an arbitrary institution on the left to the diamond-shaped core diagram for classifications on the right). Fusion or the Grothendieck construction (Grothendieck, 1963) is a way for homogeneously handling situations of structural heterogeneity (Goguen, 2006). It maps the heterogeneous situations represented by indexed contexts to the homogeneous situations represented by contexts. There are two kinds of fusion. The basic fusion process operates on indexed contexts and returns contexts. The derived fusion process has two stages: coalescence, then basic fusion. There are two versions of the general theory architecture (Fig. 23.2), one the normal version and the other the dual version. These are linked by the three involutions (see the discussion on duality below) for contexts, indexed contexts and diagrams. Because these involutions are isomorphisms, the normal and dual processes for coalescence, basic fusion and derived fusion can be defined in terms of one another. However, the dual versions have simpler definitions from more basic concepts. All notions in the architecture for the institutional approach are 2-dimensional notions, having not just links between objects but also connections between links (Fig. 23.4).

The special theory fixes the ambient context to be the context of classifications $\mathbf{V} = \mathbf{Cls}$ (This equality represents assignment of the constant context \mathbf{Cls} to the variable context \mathbf{V}). The context \mathbf{Cls} is the central context in the theories of Information Flow and Formal Concept Analysis (Ganter and Wille, 1999). The two basic ideas in the special theory are $\text{context}^{\text{core}}$ s (rectangle with lens-shaped top edge) and institutions (small triangle). The context of $\text{context}^{\text{core}}$ s is a subcontext of (core-shaped diagram within) the context of passages from \mathbf{Cls} -indexed contexts to contexts (the **Theory** node in the core diagram (Fig. 23.8) is the fusion of one example of a \mathbf{Cls} -indexed context that maps a classification to its context of theories and maps an infomorphism to its theory passage via inverse flow). Hence, the context of $\text{context}^{\text{core}}$ s is an exponential context – a product-exponent adjoint currying operator (upper corners) is used on fusion in the general theory before specializing. Connections (links of links; Fig. 23.4) in $\text{context}^{\text{core}}$ are called modifications. Institutions are \mathbf{Cls} -diagrams, diagrams in the ambient context of classifications. The basic process in the special theory is fusion: the fusion of an institution is a passage from the context of $\text{context}^{\text{core}}$ s to the context of contexts. The

core-shaped fusion diagram for an institution (left side Fig. 23.8) is embeddable into the universal core-shaped fusion diagram based only on the context of classifications (right side Fig. 23.8). However, this universal core-shaped fusion diagram is actually just one of the core-shaped fusion diagrams (left side Fig. 23.8) gotten by using the terminal institution consisting of the identity classification passage I_{Cls} . Links between embeddings is coherently connected through modifications. Although potentially large, the core diagram shape is actually quite small for practical purposes. And it is different depending on duality. Just like the general theory, there are two versions of the special theory architecture, normal and dual, and these are also linked by involutions, a diagram involution for institutions and a composite (core and context) involution for context^{core}s. The normal version (institutions and institution morphisms) has a naturally defined core diagram consisting of structures, theories and logics, plus linking and connecting elements. The dual version (institutions and institution comorphisms) has a simple core diagram consisting only of theories. The advantage of the normal version is the existence of (local) logic contexts (the **Logic** node in the core-shaped fusion diagram for an institution (left side Fig. 23.8) is the fusion of one example of a indexed context that maps a language to its logic order and maps a language morphism to its logic order map). The advantage of the dual version is the cocontinuity of theory passages (they preserve sums of theories). For either notion of fusion (normal or dual), the theory index in the core represents the lattice of theories construction as a passage from the context of institutions (normal version) or the context of coinstitutions (dual version) to the context of contexts (the context of coinstitutions has institutions as objects and institution comorphisms as links).

23.2 Contexts

23.2.1 General Theory

Both languages and classifications are objects of a mathematical context. A *mathematical context* (Fig. 23.2) corresponds to some species of mathematical structure (Goguen, 1991) (see Chapter 21, by Healy, this volume, for a more detailed discussion). It consists of a collection of objects, a collection of links relating one object to another, a way to compose links into new links, and a special identity link for each object. A link in a context (Fig. 23.3) has an orientation or direction; that is, it begins or originates at one object and ends or has destination at another. The link relates the beginning or originating object with the ending or destination object. Two links are composable when the destination of one is the origin of the other. A link in a context is an isomorphism when there is another bicomposable link, where the two compositions are identity. Any order is a context with the orderings being the links. A 2-dimensional mathematical context also has a collection of connections (Fig. 23.4), which relate one link to another link – connections are links between links. There are vertical and horizontal ways to compose connections into new

Fig 23.3 Link

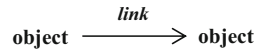
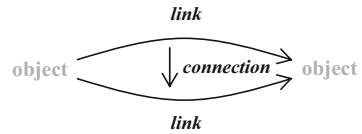


Fig. 23.4 Connection



connections. The horizontal way corresponds to link composition (see the discussion of 2-categories in Chapter 21, by Healy, this volume).

A *passage* (Fig. 23.5) from a beginning or originating context to an ending or destination context consists of a map from the beginning object collection to the ending object collection and a map from the beginning link collection to the ending link collection that preserves linkage direction and composition. Passages themselves can be composed. Two passages are composable when the destination of one is the origin of the other. The composition passage is defined coordinate-wise: compose the object maps and compose the link maps. An order map is a passage between two orders with order-preservation representing the link map. A 2-dimensional passage between 2-dimensional contexts also has a map of connections that preserves connecting direction and vertical and horizontal composition. For any map beginning in one collection and ending in another collection, a fiber over a fixed item in the destination collection is the collection of all elements in the beginning collection that map to that fixed item. Similarly, for any passage, a fiber over a fixed object in the ending context is a subcontext of the beginning context consisting of the collection of all objects that map to that fixed object and the collection of all links that map to the identity link on that fixed object. For a 2-dimensional passage, the fiber contains the collection of connections that map to the identity connection on that identity link.

A *bridge* (Fig. 23.6) from a beginning or originating passage to an ending or destination passage (between the same mathematical contexts) consists of a map from the originating context's object collection to the destination context's link collection, which naturally preserves linkage. These links in the destination context are called components of the bridge. Passages and bridges can be (horizontally) composed when the destination context of one is the originating context of the other. In the passage-bridge composition, the object that indexes a component is initially mapped by the passage. In the bridge-passage composition, the component (that is

Fig.23.5 Passage

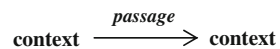
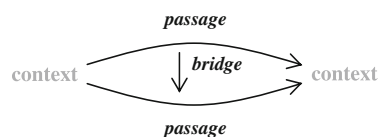


Fig. 23.6 Bridge



indexed by an object) is finally map by the passage. Bridges themselves can be composed, both vertically and horizontally. Two bridges are vertically composable when the destination of one is the origin of the other. The vertical composition bridge is defined component-wise: for each object in the originating context, compose the components in the destination context. Vertical composition is orthogonal to passage composition. Two bridges are horizontally composable when the destination context of one is the originating context of the other. The horizontal composition bridge has two equal and interchangeable definitions. One is the vertical composition of the composition of the first bridge and the originating passage of the second bridge and the composition of the destination passage of the first bridge and the second bridge. The other has a dual definition. Horizontal composition is parallel to passage composition. Any ordering on a pair of order maps is a bridge. Bridges enrich the context of contexts into a 2-dimensional mathematical context with contexts as objects, passages as links and bridges as connections.

Philosophically, the notion(s) of identity takes on several forms. Two objects in a context are equal up to identity when they are the same; they are equal up to isomorphism when they are linked by an isomorphism; and they are equal up to morphism when they are linked. Two contexts are identical when they are equal; they are isomorphic when they are linked by an isomorphism; and they are equivalent when they are linked by an equivalence.

An *invertible passage (equivalence)* or inverse pair of passages from a beginning or originating context to an ending or destination context is a pair of bicomposable passages, a left passage in the same direction and a right passage in the opposite direction, which are generalized (relaxed) inverses for each other in the sense that the compositions are identity naturally up to (iso)morphism. This means that the identity passage at the beginning context is connected to the left-right composite by a bridge called the unit, and that the right-left composite is connected to the identity passage at the ending context by a dual bridge called the counit. Unit and counit can be composed with the left and right passages. Unit and counit bridges are coherently related by two “triangle equalities” that are dual to each other: the vertical composition of the unit-left (right-unit) composite with the left-counit (counit-right) composite is the identity bridge at the left (right) passage. The context of invertible passages has contexts as objects and invertible passages as links. There are left and right projections from the context of invertible passages to the context of contexts with left being covariant and right being contravariant.

Duality (Fig. 23.7) is important in the institutional approach to ontologies, and can be confusing if not approached with some caution. In the institutional approach, there are several kinds of duality at work. The opposite of a context flips the direction of its links. The opposite of a passage has the same action, but maps a

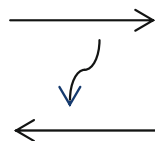


Fig. 23.7 Involution

flipped link to the flip of the image link. The opposite of a bridge has the same components, hence has its direction flipped. The opposite of an invertible passage applies the opposite to all of its components, left, right, unit and counit. It flips its direction and changes dual notions: the left (right, unit, counit) of the opposite is the opposite of the right (left, counit, unit). A contravariant passage is a (covariant) passage from the opposite of a context – same action, but change of perspective.

Duality leads to some important involutions in the institutional approach, which are defined on the basic ideas in Fig. 23.2 – contexts, indexed contexts and diagrams. The context involution maps a context, passage or bridge to its opposite. It is a 2-dimensional isomorphism on the context of contexts, which is covariant on passages and contravariant on bridges. The indexed context involution maps a dual indexed context, a dual indexed link or a dual indexed connection to itself, but changes from a contravariant to a covariant perspective. It is a 2-dimensional isomorphism between the context of dual indexed contexts and the context of indexed contexts, which is covariant on dual links and contravariant on dual connections. There is a fibered version of this: for any ambient context \mathbf{V} , there is a \mathbf{V} -indexed context involution from the context of dual \mathbf{V}^{op} -indexed contexts to the context of \mathbf{V} -indexed contexts, where \mathbf{V}^{op} is the opposite of \mathbf{V} . For any ambient context \mathbf{V} , the diagram involution maps a \mathbf{V}^{op} -diagram or \mathbf{V}^{op} -diagram colink to its opposite \mathbf{V} -diagram or \mathbf{V} -diagram link, defined by flipping its components, either indexing by context and passage or indexing passage and bridge. It is an isomorphism between the context of \mathbf{V}^{op} -codiagrams and the context of \mathbf{V} -diagrams, which is covariant on morphisms.

Several constructions beyond duality are also used in the institutional approach. The product of two collections contains all pairs of elements, where the first (second) element is from the first (second) collection. Similarly, the *product* of two contexts contains all pairs of objects and all pairs of links from the component contexts, which preserve products of origins and destinations. The two original contexts are called the components of the product. The exponent of two collections has all the maps between the collections as elements. Similarly, the *exponent* of two contexts has all the passages between the contexts as objects, the bridges between these passages as links, and vertical composition of these bridges as composition. Finally, any passage with a product origin has an *adjoint* form, which is a passage from one product component to the exponent of the other product component and the destination of the passage.

23.2.2 Special Theory

The context of classifications \mathbf{Cls} has classifications as objects and infomorphisms as links.³ A classification has instance and type collections and a classification

³The category \mathbf{Cls} of classifications and infomorphisms is the ambient category that is used for indexing in the institutional approach to ontologies. This is the category of “twisted relations” of

relation between the two. Classifications are linked by infomorphisms, which map between instance collections in the reverse direction (the instance map is said to be contravariant), map between type collections in the forward direction (the type map is said to be covariant), and require invariance of classification: a type classifies at the origin the image of an instance if and only if the image of the type classifies at the destination the instance. There are two component projection passages, instance and type, from the context of classifications to the context of sets; instance is contravariant and type is covariant.

A theory of a classification is a subset of types. For any classification, the intent of an instance is the (theory) subcollection of types that classify the instance, and dually the extent of a type is the subcollection of instances classified by that type. A theory classifies an instance of a classification when each type in the theory classifies the instance. Thus, any classification lifts to its theories; that is, any classification has an associated instance-theory classification, where types are replaced by theories. The extent of a theory is the subcollection of all instances classified by that theory; that is, the intersection of the extents of the types in the theory. A theory entails a type when any instance classified by the theory is also classified by the type; that is, when the extent of the theory is contained in the extent of the type. Theories are linked by type maps that preserve entailment. The collection of all types entailed by a theory is called the closure of the theory. Closure is an operator on theories. Types in the theory are called axioms, whereas types in the closure are called theorems. A (local) logic of a classification has two components, a theory and an instance, with a common underlying language.

For any classification, the three collections of instances, theories and logics are ordered. Two instances are ordered when any type that classifies the second instance also classifies the first instance; that is, when the intent of the first contains the intent of the second. Two theories are ordered when any axiom of the second is a theorem of the first; that is, when any type in the second theory is entailed by the first theory; that is, when the closure of the first theory contains the (closure) of the second theory; that is, when the extent of the first is contained in the extent of the second. Two logics are ordered when their instance and theory components are ordered. In any classification, the intent of an instance is maximal in subset order and minimal in entailment order. This defines a maximal theory map from instances to theories.

Give any map, direct image and inverse image form an invertible pair of maps between subsets that preserves inclusion. For any infomorphism, the (covariant) direct flow on theories is the direct image of the type function and the (contravariant) inverse flow on theories is the composition of closure followed by the inverse image of the type function. For any infomorphism, the type function preserves entailment: if any theory entails a source type at the origin, then the direct flow of the theory entails the image of the type at the destination. When the instance function of the

(Goguen and Burstall, 1992). This is also the basic category used in the theory of Information Flow and Formal Concept Analysis (Kent, 2002).

infomorphism is surjective, the type function allows borrowing (useful in theorem-proving): any theory entails a type at the origin if and only if the direct flow of the theory entails the image of the type at the destination. Any infomorphism lifts to its theories, where the type map is replaced by the direct flow map. Lifting to theories is a passage on the context of classifications. For any infomorphism, the (contravariant) logic map, which maps between logic collections in the reverse direction, has two components, the inverse flow and the instance map. For any infomorphism, the instance map preserves order on instances, the direct and inverse flow preserve (entailment) order on theories, and the logic map preserves order on logics. Direct and inverse flows form an invertible map on theories. They also form an invertible map on logics. Direct flow on logics preserves soundness; the direct image of a sound logic is also sound. Inverse flow on logics preserves completeness; the inverse image of a complete logic is also complete. Direct and inverse flow on logics forms the basis of the theory of Information Flow.

The context of classifications is fundamental in the institutional approach, and it is (co)complete. But the context of concept lattices is equivalent to it (Kent, 2002), and the context of concept orders is pseudo-equivalent to it. Hence, these also are cocomplete. A concept order (complete order with two-sided generators) consists of an order with all meets and joins, an instance set, a type set, a map that embeds instances (types) as order elements, such that any element is the join (meet) of some subcollection of embedded instances (types). When a concept order satisfies antisymmetry (isomorphic elements are identical) it is called a concept lattice. A concept morphism links concept orders. It consists of an invertible pair of order maps called the left (right) inverse, and a map linking the instance (type) collections in the opposite (same) direction, where the left (right) inverse preserves embedded instances (types). Compositions and identities are defined component-wise. The context of concept orders (lattices) has concept orders (lattices) as objects and concept morphisms as links. The context of concept lattices is a full subcontext of the context of concept orders. Each of the three contexts (classifications, concept lattices and concept orders) comes equipped with, and is definable in terms of, component projection functors. The 2-dimensional diagram consisting of these three contexts, along with their connecting passages and bridges, is called the conceptual core.

23.3 Indexed Contexts

23.3.1 General Theory

An index is a pointer used to indicate a value. A map or list is the simplest mathematical representation for an index, mapping an indexing set to a set of items of a certain type. A passage is a structured representation for an index, mapping an indexing context to a context of objects of a certain type. A structured index of a certain type X is describe as an indexed X . An *indexed context* (Fig. 23.2) is a passage into the context of contexts. As such, it is a special kind of diagram. It originates at an indexing context, maps indexing objects to component contexts, maps

indexing links to component passages, inverts direction (it is contravariant) and preserves composition and identities up to isomorphism. An indexed order is the special case of a passage into the context of orders – the passage maps indexing objects to component orders, and maps indexing links to component order maps. The involute of an indexed context is the composition of the indexed context with the context involution. A *dual indexed context* is the same; except it preserves direction (it is covariant). The indexed involution maps a dual indexed context to an indexed context by flipping the indexing context to its opposite and changing the perspective from covariance to contravariance.

An *inversely-indexed context* (corresponding to the notion of a locally reversible indexed context in Tarlecki et al. (1991)) is a passage into the context of invertible-passages. It also originates at an indexing context, mapping indexing objects to component contexts and indexing links to component invertible passages, inverting direction, and preserving composition and identities up to isomorphism. There are two components, a left component indexed context and a right component dual indexed context. We think of the modifiers “inversely” and “locally reversible” as applying to the left component indexed context; thus, inversely indexed contexts are special indexed contexts with right inverses. We define cocompleteness for dual indexed and inversely-indexed contexts. A dual index context is component-complete when all component contexts are complete. It is component-continuous when it is component-complete and all the component passages are continuous. It is cocomplete when it is component-continuous and the indexing context is cocomplete. An inversely indexed context is component-complete (component-continuous, cocomplete) when its right inverse dual indexed context is so.

An *indexed link* from a beginning or originating indexed context to an ending or destination indexed context consists of an indexing passage from the indexing context of origin to the indexing context of destination and a bridge from the beginning passage to the composition of the opposite of the indexing passage with the ending passage. Indexed links can be composed. Two indexed links are composable when the destination of one is the origin of the other. The composition of a composable pair is defined by the composition of their indexing passages and the vertical composition of their bridges. An indexed order map between indexed orders is the special case where the bridge components are order maps. A *dual indexed link* is the same, except that it links dual indexed contexts. The indexed involution maps a dual indexed link to an indexed link by flipping the indexing passage to its opposite and changing the perspective from covariance to contravariance. We define cocontinuity for only dual indexed links. A dual indexed link is component-continuous when it links component-continuous dual indexed contexts, and the component passages of its bridge are continuous. It is cocontinuous when it links cocomplete dual indexed contexts, it is component-continuous, and its indexing passage is cocontinuous.

An *indexed connection* from a beginning or originating indexed link to an ending or destination indexed link consists of an indexing bridge from the beginning indexing passage to the ending indexing passage, which preserves bridging up to morphism in the sense that the beginning bridge is linked to the vertical composition of the ending bridge with the opposite of the indexing bridge. Indexed connections can

be composed by vertical composition of their indexing bridges. A *dual indexed connection* is the same, except that it links dual indexed links. The indexed involution contravariantly maps a dual indexed connection to an indexed connection by flipping the indexing bridge to its opposite and changing the perspective from covariance to contravariance.

The context of indexed contexts is a 2-dimensional context, whose objects are indexed contexts, whose links are indexed links, and whose connections are indexed connections. The context of indexed-orders is a special case. Similar comments hold for the dual notions. The indexed involution is a 2-dimensional isomorphism from the context of dual indexed contexts to the context of indexed contexts, which is covariant on indexed links and contravariant on indexed connections. There is a 2-dimensional indexing passage from the context of indexed contexts to the context of contexts, which maps an indexed context to its indexing context, maps an indexed link to its indexing passage, and maps an indexed connection to its indexing bridge. Also, there is a 2-dimensional indexing passage from the context of indexed orders to the context of contexts.

In the institutional approach we are interested in the fibers for the 2-dimensional indexing passage from the context of indexed orders to context of contexts. The *indexed fiber* (Fig. 23.2) over a fixed context consists of the following: the objects are the indexed orders with that fixed context as indexing context, the links are the indexed order maps with the identity passage on that fixed context as indexing passage, and the connections are the indexed connections with the identity bridge on that identity passage as indexing bridge. Hence, an indexed link in a fiber, called an indexed order map, consists of a bridge between origin and destination indexed orders, and an indexed connection in a fiber, called an order pair of indexed order maps, consists of an ordering between origin and destination indexed order maps. We have inversely-indexed orders in fibers. Furthermore, we can define dual versions of all these notions. Finally, we define an additional notion in a fiber: an indexed invertible pair of order maps is a pair of bicomposable indexed order maps, a left indexed order map in the same direction and a right indexed order map in the opposite direction, which are generalized (relaxed) inverses for each other in the sense that the bridge components are invertible pairs of order maps; that is, vertical composition of the bridges is unique up to order both ways.

23.3.2 *Special Theory*

Here we discuss the core-shaped universal diagram in the special theory of the architecture. Consider the fiber of indexed orders over the fixed context **CIs**. Instance is an indexed order that maps a classification to its order of instances and maps an infomorphism to its instance order map. Inverse flow is an indexed order that maps a classification to its theory (entailment) order and maps an infomorphism to the inverse flow of its type map. Logic is an indexed order that maps a classification to its logic order and maps an infomorphism to its logic order map. Direct flow is a dual indexed order that maps a classification to its theory (entailment) order and

maps an infomorphism to the direct flow of its type map. This dual indexed order is component-complete, since for any classification the component theory order is a complete lattice. It is component-continuous, since for any infomorphism the component direct image passages are continuous, being right inverses of inverse image. The direct flow dual indexed order is cocomplete, since it is component-continuous and the indexing context of classifications is cocomplete. Inverse and direct flow form an inversely-indexed order with inverse flow as left component and direct flow as right component. This inversely-indexed order is component-complete (component-continuous, cocomplete), since its direct image right adjoint dual indexed context is so. There is a maximal theory indexed order map between the instance and inverse flow indexed orders, whose bridge has the maximal theory maps as its components. From the logic indexed order, there are two projection indexed order maps to the instance and the inverse flow indexed orders. The normal core diagram shape has three nodes or index orders (instance, inverse flow and logic) and three edges or indexed order maps (maximal theory and logic projections). Its fusion (Fig. 23.8), a core diagram in the context of contexts, has four contexts (classification, instance, theory and logic) and five passages (projections, base passages, and maximal theory). The instance projection passage within the core diagram for classifications, just like the structure projection passage within the core diagram for an institution (Fig. 23.8), is the lifting of its base passage originating from the context of theories. The dual core diagram shape has one node or dual index order (direct flow) and no edges. Its fusion is just the context of theories and the context of classifications with a base passage in between. A link in the context of instances is an infomorphism that maps the instance at the destination to a specialization of the instance at the origin. An instance is more specialized that another when it is classified by more types. A link in the context of theories is an infomorphism that maps the theory at the destination to a specialization of the theory at the origin; equivalently, maps the theory at the origin to a generalization of the theory at the destination. A theory is more specialized that another when any axiom of the second

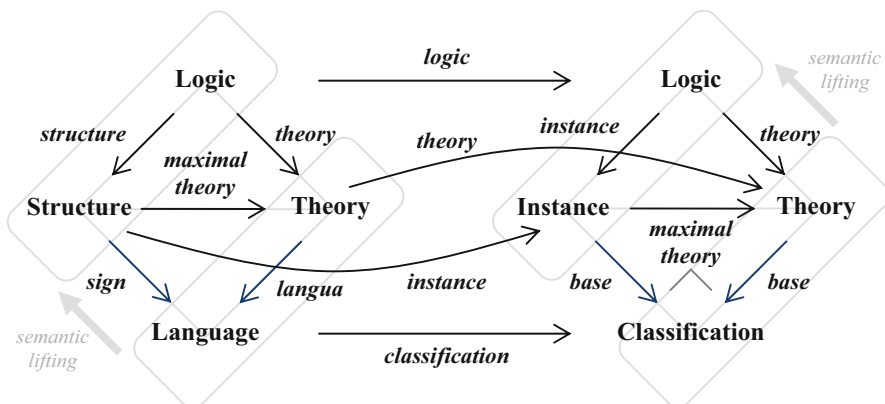


Fig. 23.8 Logical map between core diagrams

is a theorem of the first. A link in the context of logics is an infomorphism that is both an instance link and a theory link.

23.4 Diagrams

23.4.1 General Theory

Let \mathbf{V} be any context within which we will work. Of course, one would normally choose a context \mathbf{V} that has some useful properties. We keep that context fixed throughout the discussion and call it the ambient context. We regard the objects and links in the ambient context \mathbf{V} to be values that we want to index, and we focus on a particular part of the ambient context \mathbf{V} . We use a passage into \mathbf{V} for this purpose. A *diagram* (Fig. 23.9) is a passage from an indexing context into the ambient context \mathbf{V} . The objects in the indexing context are called indexing objects and the links are called indexing links. A diagram may be presented by using a directed graph to generate an indexing path context. If so, such a graph is usually called the shape of the diagram. By extension, the indexing context of any diagram can be called its shape. The diagram involution maps a \mathbf{V}^{op} -diagram to its opposite \mathbf{V} -diagram, defined by flipping its indexing context and passage.

Diagrams can be linked. A *diagram link* (Fig. 23.10) from a beginning or originating diagram to an ending or destination diagram consists of a passage between indexing contexts called an indexing passage and a bridge from the composition of the indexing passage with the destination passage to the beginning passage. Two diagram links are composable when the destination of one is the origin of the other. The composition diagram link is defined coordinate-wise: compose the indexing passages and vertically compose the bridges. Diagrams can also be linked in a dual fashion. A *diagram colink* from a beginning or originating diagram to an ending or destination diagram consists of an indexing passage as before, but a bridge in the opposite direction: from the beginning passage to the composition of the indexing passage with the destination passage. Composition is defined similarly. The diagram involution maps a \mathbf{V}^{op} -diagram colink to its opposite \mathbf{V} -diagram link, defined by

Fig. 23.9 Diagram

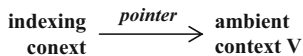
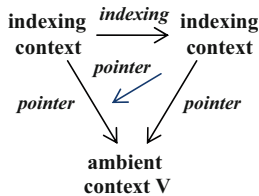


Fig. 23.10 Diagram link



flipping its indexing passage and bridge. There is a context of \mathbf{V} -diagrams with diagrams as objects and diagram links as links, and there is a context of \mathbf{V} -codiagrams with diagrams as objects and diagram colinks as links. There is an indexing passage to the context of contexts from the context of \mathbf{V} -(co)diagrams, which maps a diagram to its indexing context and maps a diagram (co)link to its indexing passage. The diagram involution is an isomorphism between the context of \mathbf{V}^{op} -codiagrams and the context of \mathbf{V} -diagrams, which is covariant on morphisms. There is a terminal diagram consisting of the identity passage on the ambient context \mathbf{V} , so that \mathbf{V} is the indexing context. From any diagram there is a trivial diagram (co)link to the terminal diagram with the indexing passage being the passage of the diagram and the bridge being identity.

Although dual, the links and colinks between diagrams seem to be independent. However, a strong dependency exists when their indexing passages are invertible. Such a strongly dependent pair is called a duality. More precisely, a *duality* is a pair consisting of a diagram colink and a diagram link whose indexing passages form an invertible pair of passages with the left component the indexing passage for the diagram colink and the right component the indexing passage for the diagram link. This implies that colink and link are between diagrams in opposite directions. Then colink and link are definable in terms of each other: (loosely) the bridge of the colink is the vertical composition of the unit with the bridge of the link and the bridge of the link is the vertical composition of the bridge of the colink with the counit.

In the institutional approach we are interested in the fibers for the indexing passage from the context of \mathbf{V} -codiagrams to context of contexts. The indexed fiber over a fixed context consists of the following: the objects are the \mathbf{V} -diagrams with that fixed context as indexing context, and the links are the \mathbf{V} -diagram colinks with the identity passage on that fixed context as indexing passage. If we think of the indexing passage as a means of moving diagrams along indexing contexts, then a fiber link is one with an identity indexing passage. That is, a fiber link is just a bridge between two passages with the same shape that map into the ambient context \mathbf{V} ; it is a bridge from the passage of origin to the passage of destination. A constant diagram is a diagram that maps all indexing objects to a particular object in \mathbf{V} and maps all indexing links to the identity on that object of \mathbf{V} . For any object in \mathbf{V} and any context (as shape), there is a constant diagram over that object with that shape. A *corelation* is a fiber link to a constant destination diagram. A constant diagram link is a fiber link between two constant diagrams (with the same shape) whose bridge components are all the same – a particular link in \mathbf{V} between the objects of the constant diagrams. For any link in \mathbf{V} and any context (as shape), there is a constant diagram link over that link with that shape.

For any diagram, a summing corelation is an initial corelation originating from that diagram: any other corelation originating from that diagram is the vertical composition of the summing corelation with the constant diagram link over a unique link in \mathbf{V} . Any two summing corelations for the same diagram are isomorphic, and hence conceptually identical. For a summing corelation the object is called a *sum* of the diagram and the component links are called sum injections. The sum of a diagram is a kind of constrained sum: disjointly sum the component objects indexed by

the diagram, and constrain these objects with the links indexed by the diagram. The ambient context \mathbf{V} is said to be *cocomplete* when sums exist for all diagrams into it. Relations, producing relations, products and completeness are defined dually. Two equivalent contexts have the same sums up to isomorphism – a sum in one context is isomorphic to a sum in an equivalent context. Two pseudo-equivalent contexts have the same sums up to equivalence – a sum in one context is equivalent to a sum in a pseudo-equivalent context. The context of \mathbf{V} -diagrams is complete (cocomplete) when \mathbf{V} is complete (cocomplete), and the context of \mathbf{V} -codiagrams is complete (cocomplete) when \mathbf{V} is cocomplete (complete) (Goguen and Roşu, 2002). Hence, the context of \mathbf{V}^{op} -codiagrams is complete (cocomplete) when \mathbf{V}^{op} is complete (cocomplete). This is compatible with the fact that the context of \mathbf{V}^{op} -codiagrams is isomorphic to the context of \mathbf{V} -diagrams via the diagram involution. There is an issue about the “smallness” of diagrams that we are ignoring here. Composition by a passage maps a diagram in the originating context (an originating diagram) to a diagram in the destination context (a destination diagram). A passage is *cocontinuous* when the passage maps the sum of any originating diagram to a sum of the corresponding destination diagram. Continuity of a passage has a dual definition using limits.

23.4.2 Special Theory

An *institution* or logical system (Fig. 23.11) is a diagram in the ambient context of classifications \mathbf{Cls} . The context of institutions is the context of \mathbf{Cls} -diagrams, where links are called institution morphisms (Fig. 23.12). The context of coinstitutions is the context of \mathbf{Cls} -codiagrams, where links are called institution comorphisms. There is a terminal institution, which is the terminal \mathbf{Cls} -diagram; it consists of the identity classification passage $I_{\mathbf{Cls}}$. The ambient context \mathbf{Cls} is both complete and cocomplete. Hence, the context of institutions and the context of coinstitutions are both complete and cocomplete, with the identity passage on \mathbf{Cls} being the terminal institution. (Much of the theory of Information Flow is based upon the fact that \mathbf{Cls} is cocomplete – the distributed systems of Information Flow are represented

Fig. 23.11 Institution

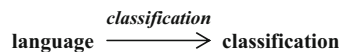
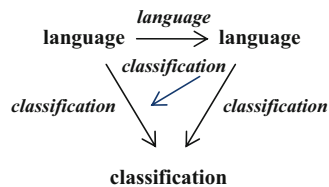


Fig. 23.12 Institution morphism



as diagrams in **CIs**, channels covering such distributed systems are represented as corelations in **CIs**, and minimal covering channels are represented as sums in **CIs**.)

In more detail, an institution (Goguen and Burstall, 1992) consists of an indexing language context and a classification passage into the ambient context of classifications. Indexing objects for an institution are called languages or vocabularies, and indexing links are called language or vocabulary morphisms. The classification passage maps a language to a classification called the “truth classification” of that language (Barwise and Seligmann, 1997). For any language, the instances of its classification are called structures, the types are called sentences, and the classification relation is called satisfaction. For any structure and sentence in a satisfaction relationship, we say that “the structure satisfies the sentence” or “the sentence holds in the structure” or “the sentence is satisfied in the structure” or “the structure is a structure of the sentence”. The classification passage maps a language morphism to an infomorphism, and the invariance of classification expresses the invariance of truth under change of notation: the image of a structure satisfies a sentence at the origin if and only if the structure satisfies the image of the sentence at the destination. A structure satisfies (is a model of) a theory when it satisfies all axioms of (sentences in) the theory. A theory entails a sentence when that sentence is true in all models of the theory. The closure of a theory is the collection of sentences entailed by the theory. Two theories are ordered by entailment when the first theory entails every axiom of the second theory; that is, when the closure of the first theory contains the second theory. Two models are entailment ordered when the theories that they induce are so ordered. Two logics are entailment ordered when their component models and theories are so ordered.

The equivalences between the context of classifications and the context of concept lattices or the context of concept preorders allows either of these to be equivalently used as the ambient context in defining the context of (co)institutions (Kent, 2004). That is, we can equivalently regard an institution to be a diagram of concept lattices. Here each language indexes a concept lattice, where the intent of a concept is a closed theory and concept order is reversed subset order on closed theories. Or, we can equivalently regard an institution to be a diagram of concept preorders. Here each language intentionally indexes a “lattice of theories” with each preorder element being a theory and order on theories being entailment order.

Examples of institutions include the following (Goguen and Burstall, 1992; Goguen and Roşu, 2002; Goguen, 2007): first order logic with first order structures as structures, many sorted equational logic with abstract algebras as structures, Horn clause logic, and variants of higher order and of modal logic. Other examples (Mossakowski et al., 2005) of institutions include intuitionistic logic, various modal logics, linear logic, higher-order, polymorphic, temporal, process, behavioral, coalgebraic and object-oriented logics. Here are more detailed descriptions of some institutions.

In the institution **EQ** of equational logic (universal algebra), a language is a family of sets of function symbols, and a language morphism is a family of arity-preserving maps of function symbols. The set of sentences indexed by a language is the set of equations between terms of function symbols. The sentence translation function

indexed by language morphism is defined by function symbol substitution. A structure of equational logic is an algebra, consisting of a set (universe) and a function for each function symbol in the language. Structure translation is a substitution passage; it is reduct with symbol translation. Satisfaction is as usual.

The institution **FOL** of unsorted first-order logic with equality extends the institution of equational logic by adding relation symbols. A language is a family of sets of function symbols as above, plus a family of sets of relation symbols with arity. A language morphism is a family of arity-preserving maps of function symbols as above, plus a family of arity-preserving maps of relation symbols. Sentences are the usual first order sentences. The set of sentences indexed by a language consists of closed first order formulae using function and relation symbols from the language. The sentence translation function indexed by a language morphism is defined by symbol substitution. Structures are the usual first order structures. A structure is a set (universe), a function for each function symbol (that is, an algebra, as above), and for each relation symbol a subset of tuples of that arity. Structure translation is reduct (substitution) with symbol translation. Satisfaction is as usual.

The theory of sketches (Barr and Wells, 1999) is a categorical approach for specifying ontologies. A sketch signature is a graph plus collections of cones and cocones each with an arity base in that graph. In an interpretation of a sketch the nodes of the underlying graph are intended to specify sorts (or types), the edges are intended to specify algebraic operations, and the cones (cocones) are intended to specify products (sums). A morphism between sketch signatures is a graph morphism between the underlying graphs of origin and destination, which preserves arity by mapping source cones (cocones) to target cones (cocones). Any context has an underlying sketch signature, whose graph is the underlying graph of the context (nodes are objects and edges are links), and whose cones (cocones) are the limiting cones (colimiting cocones) in the context. Given a sketch signature, a diagram in that signature is a diagram in the underlying graph; in a diagram any pair of paths with common beginning and ending nodes is called an equation. Given a sketch signature, an interpretation of that sketch signature in a fixed context (the context of finite sets is used in Johnson and Rosebrugh, 2007) is a sketch signature morphism into the underlying signature of the context, a graph morphism that maps cones (cocones) in the sketch to limiting cones (colimiting cocones) in the context. An interpretation morphism is a graph bridge from one interpretation to another. Given a sketch signature, an interpretation and a diagram in that signature, the interpretation satisfies the diagram when it maps the diagram to a commutative diagram in the fixed context; a commutative diagram in a context is a diagram where the composition of each side of an equation is equal.

A sketch consists of a sketch signature plus a collection of diagrams (equivalently, equations) in the underlying graph. The diagrams are intended to specify commutative diagrams in an interpretation. For any interpretation and sketch having the same signature, the interpretation satisfies the sketch, and is called a model (or an algebra) of the sketch, when it satisfies every diagram in the sketch. A homomorphism of models (algebras) is an interpretation morphism between models. Any interpretation defines a sketch (for which it is a model) having the same underlying

signature and consisting of all diagrams satisfied by the interpretation. A sketch entails a diagram in its signature when any model of the sketch satisfies that diagram. Two sketches are ordered by entailment when they have the same underlying signature and any diagram of the second sketch is entailed by the first sketch. This is a preorder (reflexive and transitive). The closure of a sketch is another sketch with the same signature, but which contains all entailed diagrams. This is a closure operator on reverse entailment order (monotonic, increasing and idempotent). A sketch and its closure are isomorphic w.r.t. entailment order. A sketch is closed when it is identical to its closure. A sketch morphism is a sketch signature morphism, which maps entailed diagrams of the first sketch to entailed diagrams of the second sketch.

In the institution \mathbf{Sk} of sketches, a language is a sketch signature and a language morphism is a sketch signature morphism. The set of sentences indexed by a sketch signature is the set of diagrams in that signature. The sentence translation function indexed by a sketch signature morphism maps source diagrams to target diagrams by graph morphism composition. The set of structures indexed by a sketch signature is the set of interpretations of that signature. The structure translation function indexed by a sketch signature morphism is reduct (substitution). Satisfaction is defined above. Given any sketch signature morphism, the invariance of satisfaction expresses the invariance of truth under change of notation: the reduct of an interpretation satisfies a diagram at the origin if and only if the interpretation satisfies the translation of the diagram at the destination. There are several special kinds of sketch signatures useful for particular purposes: linear sketch signatures (or graphs) with neither cones nor cocones used, product sketch signatures (or multi-sorted equational logic languages) with only discrete cones and no cocones used, limit sketch signatures with any cones but no cocones used, and limit-coproduct sketch signatures (see [Chapter 24](#), by Johnson and Rosebrugh, this volume) with any cones but only discrete cocones used. Each of these special kinds of sketch signatures forms its own institution. In general, there is a trivial inclusion institution morphism from any one of these kinds to a more powerful kind; for example, from limit sketch signatures to general sketch signatures. Often in these special sketches, the (co)cones and graphs are required to be finite. The notion of a limit-product sketch is used to define the entity-attribute data model in (Johnson and Rosebrugh, 2007), which is an enriched extension of the traditional entity-relationship-attribute data model. The paper (Johnson and Rosebrugh, 2007) requires the model reduct passage to be a (op)fibration in order to define universal view updatability; a notion of cofibration is defined on sketch morphisms to ensure that this holds. It is an interesting question whether these notions have meaning and importance for an arbitrary institution.

Information systems for any institution of sketches can be defined over the context of sketches (theories), the context of models or algebras (sound logics), or even over a larger context of logics (not defined here). Within this institution a theory is a sketch and a theory morphism is a sketch morphism. This defines the context of theories. From this context there is an underlying passage to the context of sketch signatures. Any sketch signature has an associated context of interpretations and

their morphisms. Any sketch signature morphism has an associated reduct (substitution) passage from target fiber context of interpretations to source fiber context of interpretations; this is defined by composition of the sketch signature morphism with interpretations and their morphisms. Hence, there is an indexed context from the context of sketch signatures (languages). The Grothendieck construction on this indexed context forms the context of structures (interpretations): an object is a pair consisting of a sketch signature and an interpretation having that signature; and a morphism is a pair consisting of a sketch signature morphism and a graph bridge from the originating interpretation to the reduct of the destination interpretation. From this context there is an underlying language passage (split fibration) to the context of sketch signatures. Any sketch has an associated a context of models (algebras) and their homomorphisms. For any sketch morphism, structure translation (reduct) preserves model satisfaction. Thus, any sketch morphism has an associated reduct passage between fiber contexts of models. Hence, there is an indexed context from the context of sketches (theories). See the model category functor in (Barr and Wells, 1999). Fusion (the Grothendieck construction) on this indexed context forms the context of sound logics (models or algebras): an object is a pair consisting of a sketch and a model (algebra) of that sketch; and a morphism is a pair consisting of a sketch morphism and a homomorphism from the originating model to the reduct of the destination model. From this context there are projection passages to the context of sketches and the context of interpretations.

A harmonious unification between the theories of institutions and Information Flow works best in a logical environment. A *logical environment* is a structured version of an institution, which takes the philosophy that semantics is primary. A logical environment requires a priori (1) the existence of a context of structures that is cocomplete and (2) the existence of a fibration (Cartesian passage) from the context of structures to the context of languages that factors through the order-theoretic and flatter context of structures built by fusion from just the underlying institution. An even more structured version of logical environment requires existence of a left adjoint to the fibration. The basic institution of Information Flow and an analogous institution of sorted first order logic are important examples of such logical environments.

The logical environment **IFC** is the basic institution for Information Flow. A structure is a classification, and a structure morphism is an infomorphism. A language is a set (of types) and a language morphism is a (type) function. The underlying language passage from the context of classifications to the context of sets is the type projection passage. A sentence is a sequent of types consisting of pairs of type subsets, antecedent and consequent. Sentence translation is direct image squared on types. A theory consists of sets of sequents; equivalently, a theory consists of a (type) set and an endorelation on subsets of types. A closed theory is known as a regular theory in Information Flow (satisfies for example, identity, weakening and global cut). A theory morphism is a (type) function maps source sequents into the target closure. When the target theory is closed, a theory morphism maps source sequents to target sequents. A classification satisfies a sequent when any instance classified

by all types in the antecedent is classified by some type in the consequent. The logical environment **IFC** is a subenvironment of the first order logical environment **FOL** when types are regarded as unary relation symbols.

23.5 Coalescence

Based upon the horizontal composition of passages and bridges, there is a composition called *coalescence* (Fig. 23.2) from the context product of **V**-diagrams and **V**-indexed contexts to the context of indexed contexts. There is also a dual version of coalescence from on the context product of **V**^{OP}-codiagrams and dual **V**^{OP}-indexed contexts to the context of dual indexed contexts. By using the involution on indexed contexts and the involution on diagrams, these two versions of coalescence can be defined in terms one another: the dual coalescence followed by the indexed context involution is the same as the passage product of the involutions for **V**-diagrams and **V**-indexed contexts followed by coalescence.

However, the dual version of coalescence can be defined directly in terms of horizontal composition. For any ambient context **V**, in the context of codiagrams an object is essentially a passage and a morphism contains a bridge, both with destination context **V**. Also, in the context of dual indexed contexts an object is essentially a passage and a morphism is essentially a bridge, both with originating context **V**. Hence, horizontal composition can be applied to both. The coalescence of a **V**-diagram and a dual **V**-indexed context is a dual indexed context, whose indexing context is that of the diagram and whose passage is the composition of component passages. The coalescence of a **V**-diagram colink and a dual **V**-indexed link is a dual indexed link, whose indexing passage is that of the **V**-diagram colink and whose bridge is the horizontal composition of component bridges.

23.6 Fusion

23.6.1 General Theory

Following the paper (Goguen, 2006), *fusion*⁴ (Fig. 23.2) or the Grothendieck construction (Grothendieck, 1963) is a way for homogeneously handling situations of structural heterogeneity. Such situations are represented by indexed contexts, where one kind of structure is indexed by another, are a central structure found in the institutional approach to ontologies. The fusion process transforms by structural summation an indexed context into a single all-encompassing context. There are two

⁴The fusion of an indexed context might be called “fusion in the large” or “structural fusion”. The sums of diagrams, in particular sums of diagrams of theories of an institution, which takes place within the fused context of theories, might be called “fusion in the small” or “theoretical fusion”. Both are kinds of constrained sums.

zones involved in fusion: the zone of indexing and the zone of structural summation. The zone of indexing often contains pseudo notions, where structure is unique only up to isomorphism. However, the zone of structural summation is strict. Links in the fused context can represent sharing and translation between objects in the structurally heterogeneous indexed context, and indexed objects in the fused context can be combined using the sum construction. Examples include ontologies.

In overview, the component contexts of an indexed context are assembled together by fusion into a single homogeneous context obtained by forming the disjoint union of their object collections and then adding links based on the indexing links. In detail, given an indexed context, define the fusion context as follows: objects are pairs called indexed objects, which consist of an indexing object and an object in the corresponding component context; and links from one indexed object to another indexed object are pairs called indexed links, which consist of an indexing link and a component link from the object of the beginning indexed object to the contravariantly mapped image of the object of the ending indexed object. Composition of indexed links is defined in terms of composition of the underlying indexing links and component links. The fusion of a dual indexed context is the opposite of the fusion of the corresponding indexed context under index context involution. For any indexed context, the fusion of the indexed context involute has the same indexed objects, but has indexed links whose component link is flipped. The fusion of an inversely indexed context is the fusion of the left component indexed context involute or, by the definition of invertibility, the fusion of the right component dual indexed context. An (dual, inversely) indexed context has a base projection passage from its fusion context to its indexing context. The fibers of an indexed context are the component contexts, whereas the fibers of a dual (inversely) indexed context are the opposite of the component contexts.⁵ The fusion context of a cocomplete dual or inversely indexed context is cocomplete, and its base projection passage is cocontinuous.

The fusion of an indexed link is a passage from the fusion of the originating indexed context to the fusion of the destination indexed context. It maps an indexed object by applying the indexing passage to its indexing object and applying the bridge to its component object, and it maps an indexed link by applying the indexing passage to its indexing link and applying the bridge to its component link. The fusion of an indexed link commutes with its indexing passage through the base projection passages of the origin and destination indexed contexts. The fusion of an indexed connection from one indexed link to another indexed link is a bridge from the fusion passage of the beginning indexed link to the fusion passage of the ending indexed link. The indexing link of the components of the fusion bridge is obtained by applying the bridge of the indexed connection to the indexing objects in the fusion of the beginning indexed context. Hence, the basic fusion process is a 2-dimensional passage from the context of (dual) indexed contexts to the context of

⁵This explains the flip in the definition of cocompleteness and cocontinuity for dual (inversely) indexed contexts.

contexts. The ordinary and dual versions are definable in terms of one another via the involutions for contexts and indexed contexts. The fusion passage of a cocontinuous dual indexed link is cocontinuous.

For any ambient context \mathbf{V} , the derived fusion process is the composition of coalescence with basic fusion. As a result, it maps a \mathbf{V} -diagram and an (dual) \mathbf{V} -indexed context to the context that is the fusion of their coalescence. It maps a \mathbf{V} -diagram (co)link and an (dual) \mathbf{V} -indexed link to the passage that is the fusion of their coalescence. There is an adjoint derived fusion process that maps a \mathbf{V} -diagram to a process that maps a (dual) \mathbf{V} -indexed context to the fusion context. The same process applies to links. Here, the (dual) \mathbf{V} -indexed contexts and links are usually restricted to a relevant subcollection called the core.

23.6.2 *Special Theory*

In the institutional approach, unpopulated ontologies are represented by theories, and populated ontologies are represented by (local) logics. Structures provide an interpretative semantics for languages, theories provide a formal or axiomatic semantics for languages, and logics provide a combined semantics, both interpretative and axiomatic. Theories are the most direct representation for ontologies. The notion of a (local) logic in an institution generalizes the notion of a (local) logic in the theory of Information Flow. Hence, for institutions and institution morphisms the relevant core indexed contexts are structure, inverse flow and logic. The fusions of these comprises the core diagram (Fig. 23.8) consisting of the context of structures representing interpretative semantics, the context of theories representing formal or axiomatic semantics and the context of logics representing combined semantics, respectively. Since the identity passage on classifications can be regarded as a terminal institution, the core diagram for classifications is just the very special case of the core diagram for this identity institution. The core diagram for an institution is linked to the core diagram for classifications by the classification passage and others built upon it. Each institution morphism generates a link between the core diagrams at origin and destination, which consists of passages for structures, theories and logics. For institutions and institution comorphisms there is only one relevant indexed context: direct flow. The fusion of direct flow is the same theory context given by inverse flow, since inverse and direct flow are components of the same inversely indexed context.

In the institutional approach to ontologies the “lattice of theories” construction is represented as the theory passage from the context of (co)institutions to the context of contexts. For any fixed institution the “lattice of theories” construction is represented “in the large” by the context of theories, and for any language of that institution the “lattice of theories” construction is represented “in the small” by either the concept preorder of theories or the concept lattice of closed theories. From each theory in the order of theories, the entailment order defines paths to the more generalized theories above and the more specialized theories below. There are

four ways for moving along paths from one theory to another (Sowa, 2000): contraction, expansion, revision and analogy. Any theory can be contracted or reduced to a smaller, simpler theory by deleting one or more axioms. Any theory can be expanded by adding one or more axioms. A revision step is composite – it uses a contraction step to discard irrelevant details, followed by an expansion step to added new axioms. Unlike contraction, expansion or revision, which move to nearby theories in the order, analogy jumps along a language link to a remote theory in the context (in a first order logic institution this might occur by systematically renaming the entity types, relation types, and constants that appear in the axioms). By repeated contraction, expansion and analogy, any theory can be converted into any other. Multiple contractions would reduce a theory to the empty theory at the top of the lattice. The top theory in the lattice of theories is the closure of the empty theory – it contains only tautologies or logical truths; i.e. sentences that are true in all structures (it is “true of everything”). Multiple expansions would reduce a theory to the full inconsistent theory at the bottom of the lattice. The full inconsistent theory is the closed theory consisting of all expressions; i.e. expressions that are true in no structures (it is “true of nothing”).

The context of theories, which is the fusion of the inverse-direct flow, is cocomplete, since inverse-direct flow is a cocomplete inversely **CIs**-indexed order (Tarlecki et al., 1991). An institution is said to be cocomplete when its direct flow coalescence, a dual indexed context, is cocomplete; equivalently, when its context of languages is cocomplete. For any institution, the projection passage reflects sums. Hence, if the institution is cocomplete, then its context of (closed) theories is cocomplete and its projection passage is cocontinuous (Goguen and Burstall, 1992). When ontologies are represented by theories (formal or axiomatic representation) in a cocomplete institution, semantic integration of ontologies can be defined via the two steps of alignment and unification⁶ (Kent, 2004). When ontologies are represented by (local) logics (interpretative and axiomatic representation) in a logical environment, semantic integration of ontologies can be defined by an analogous process (see the lifting in Fig. 23.8). An institution colink is cocontinuous when its direct flow coalescence, a dual indexed link, is cocontinuous; equivalently, when it links cocomplete institutions and its language passage is cocontinuous. If an institution colink is cocontinuous, then its theory passage is cocontinuous. For any institution duality (institution colink and link) based on an invertible passage of languages, there is a (closed) invertible passage of theories, with right (left) component being the fusion of the institution (co)link. Hence, the left (closed) theory passage is cocontinuous.

According to the dictionary, a cosmos is an orderly harmonious systematic universe. A polycosmos (Patrick Cassidy) is an unpopulated modular object-level “ontology that has a provision for alternative possible worlds, and includes some alternative logically contradictory theories as applying to alternative possible worlds”. The mathematical formulation of polycosmic is given in terms of the

⁶See Footnote 2

sum of a diagram of theories for some institution. A diagram of theories is monocosmic when its sum is consistent (satisfiable by some structure). A diagram of theories is pointwise consistent when each indexed theory in the direct flow along the summing correlation is consistent. A monocosmic diagram of theories is pointwise consistent by default. A diagram of theories is polycosmic when it is pointwise consistent, but not monocosmic; that is, when there are (at least) two consistent but mutually inconsistent theories in the direct flow. In some institutions, there are some extreme polycosmic diagrams of theories, where any two theories are either entailment equivalent (isomorphic) or mutually inconsistent. Each of the theories in these diagrams lies at the lowest level in the lattice of theories, strictly above the bottom inconsistent theory containing all sentences.

23.7 Formalism

The Information Flow Framework (IFF) (Kent et al., 2001–2007) is a descriptive category metatheory under active development, which was originally offered as the structural aspect of the Standard Upper Ontology (SUO) and is now offered as a framework for all theories. The IFF architecture is a two dimensional structure consisting of metalevels (the vertical dimension) and namespaces (the horizontal dimension). Within each level, the terminology is partitioned into namespaces. In addition, within each level, various namespaces are collected together into meaningful composites called meta-ontologies. The IFF is vertically partitioned into the object level at the bottom, the supra-natural part or metashell at the top, and the vast intermediate natural part. The natural part is further divided horizontally into pure and applied aspects. The pure aspect of the IFF is largely concerned with category-theoretic matters. The applied aspect of the IFF is largely governed by the institutional approach to the theory and application of ontologies.

The IFF has had two major developmental phases: experiment and implementation. The experimental phase of the IFF development occurred during the years 2001–2005. The present and future development is mainly concerned with the final coding and the implementation of the IFF. Initially, the plan of development was for the IFF to use category theory to represent various aspects of knowledge engineering, but more recently this strategy was augmented and reversed, thus applying knowledge engineering to the representation of category theory. The institutional approach is the main instrument used by the IFF to connect and integrate axiomatizations of various aspects of knowledge engineering. It is being axiomatized in the upper metalevels of the IFF, and the lower metalevel of the IFF has axiomatized various institutions in which the semantic integration of ontologies has a natural expression as the sum of theories.

Both semantics and formalisms are important for ontologies. The connection between semantics and formalism is through interpretation. The institutional approach is centered on interpretation and represents it as a parameterized relation of satisfaction between semantics and formalism. Although in many common

examples the formal side of the satisfaction relation is set-theoretically small and the semantical side is set-theoretically large, in the IFF axiomatization of the institutional approach both sides can range through the hierarchy of metalevels. Hence, we think of the institutional approach as existing at a higher level, with its domain including the triad (semantics, formalism, interpretation) and its formal system encoded in category-theoretic terms. However, it incorporates category theory also, with the contents of category theory as its semantics and the axiomatization of category theory (as done in the IFF) as its formal system. So category theory provides a formalism for the institutional approach, and the institutional approach provides an interpretation for category theory.

References

- Barr, M., and C. Wells. 1999. *Category theory for computing science*. Englewood Cliffs, NJ: Prentice-Hall.
- Barwise, J., and J. Seligman. 1997. *Information Flow: Logic of Distributed Systems*. Cambridge, MA: Cambridge University Press; *Tracts in Theoretical Computer Science* 44:274.
- Johnson, M., and R. Rosebrugh. 2007. Fibrations and universal view updatability. *Theoretical Computer Science* 388:109–129.
- Kent, R.E. 2002. Distributed conceptual structures. Proceedings of the 6th International Workshop on Relational Methods in Computer Science (RelMiCS 6). Lecture Notes in Computer Science 2561, Berlin: Springer.
- Kent, R.E. 2004. Semantic integration in the information flow framework. Dagstuhl Workshop on Semantic Interoperability and Integration. Sept 19–24, Seminar N 04391.
- Kent, R.E., J. Farrugia, M. Schorlemmer, and L. Obrst. 2001–2007. The Information Flow Framework (IFF). Technical reports.
- Ganter, B., and R. Wille. 1999. *Formal concept analysis: Mathematical foundations*. Springer: Berlin.
- Goguen, J. 1991. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1): 49–67.
- Goguen, J. To appear. 2006. Information integration in institutions. In *Jon Barwise Memorial Volume*, ed. L. Moss, Bloomington, IN: Indiana University Press.
- Goguen, J. 2007. Ontotheology, ontology, and society. *International Journal of Human Computer Studies*. Special issue on ontology. ed. Christopher Brewster and Kieron O’Gara. doi:10.1016/j.ijhcs.2007.04.003.
- Goguen, J., and R. Burstall. 1992. Institutions: Abstract structure theory for specification and programming. *Journal of the Association for Computing Machinery*. 39(1):95–146.
- Goguen, J., and G. Roşu. 2002. Institution morphisms. *Formal aspects of computing* 13:274–307.
- Grothendieck, A. 1963. Catégories fibrées et descente. In *Revêtements étales et groupe fondamental*, Séminaire de Géométrie Algébrique du Bois-Marie 1960/61, Exposé VI, Institut des Hautes Études Scientifiques, Paris, 1963; reprinted in *Lecture Notes in Mathematics* 224, Springer-Verlag, 1971, 145–194.
- Mossakowski, T., J. Goguen, R. Diaconescu, and A. Tarlecki. 2005. What is a logic? In J.-Y. Beziau, ed. *Logica Universalis*. 113–133. Basel: Birkhauser.
- Schorlemmer, M., and Y. Kalfoglou. 2008. Institutionalising ontology-based semantic integration. To appear (2008) in *Applied Ontology*. IOS Press.
- Sowa, J. 2000. *Knowledge representation: Logical, philosophical and computational foundations*. Pacific Grove, CA: Brooks/Coles.
- Tarlecki, A., R. Burstall and J. Goguen. 1991. Some fundamental algebraic tools for the semantics of computation, Part 3: Indexed Categories. *Theoretical Computer Science* 91:239–264.

Chapter 24

Ontology Engineering, Universal Algebra, and Category Theory

Michael Johnson and Robert Rosebrugh

24.1 Introduction

Ontology engineering is most often used to refer to the (indeed vitally important) process of constructing ontologies.

Thus it was once with traditional engineering – engineers were concerned with building essentially unique artifacts: bridges, roads and buildings for example. The engineer had learnt from other examples and would bring general concepts, known solutions, rules-of-thumb, and scientific calculation to the particular problems presented by a new site. Sometimes the engineer could effectively reuse a design, adjusting only a few parameters (height of the towers, length of a span etc) because a new site was sufficiently like an old site, but essentially each new artifact called for a newly engineered solution. In contrast much modern engineering is fundamentally about developing interoperations among extant systems – telecommunications is almost by definition thus, and most modern manufacturing depends very significantly on planning and managing the interactions between known systems.

The irony of ontology engineering needing to focus on constructing individual ontologies (whether small and domain specific (see examples in Bontas and Christoph (2006)) or extensive and intended to establish standards over a wider field (BizDex)) is of course that ontologies themselves were introduced to aid system interoperability. A good theory and detailed processes for *ontology interoperability* will significantly aid the development of new ontologies incorporating old ones. Then extant ontologies can be used to support systems interoperation even when the distinct systems are based themselves in separately developed ontologies. This was the goal of the Scalable Knowledge Composition group's *algebra of ontologies* (Mitra and Wiederhold, 2004).

M. Johnson (✉)

Department of Computer Science, Macquarie University, Sydney, Australia

e-mail: mike@ics.mq.edu.au

Research partially supported by grants from the Australian Research Council and NSERC Canada.

This chapter follows and further develops (Johnson and Dampney, 2001) in drawing ideas from categorical universal algebra, a field which might be viewed as ontology engineering for mathematics, and using them to support ontology engineering in a manner that leads naturally, and mathematically, to support for interoperations among ontologies.

The ideas presented here have a strong theoretical foundation in the branch of mathematics called *category theory* (see Chapter 21, by Healy, this volume), and they have been developed and tested in a range of industrial applications over the last 15 years.

24.2 Representing Ontologies

Formal ontologies are at least “controlled vocabularies” and can take many forms ranging from glossaries to first order logical theories (Smith and Welty, 2001). Often ontologies are expressed as trees or directed graphs because *subsumption* is such an important (real-world) relationship and we are used to representing subsumption relationships as directed edges. In fact, subsumption is a particular example of a function (to each instance of the subclass, there corresponds a particular instance in the superclass), and graphically or not, ontologies depend fundamentally on being able to represent functions.

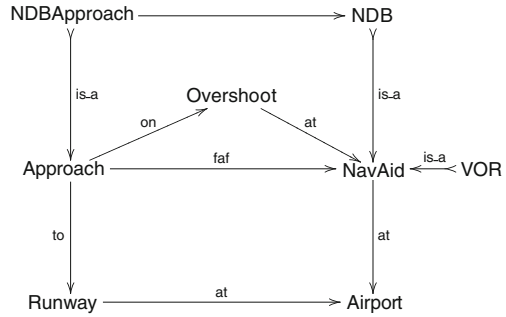
For us, it is largely unimportant how we represent ontologies because there is substantial software support for ontology engineering. It is vital that a proposed ontology is given via a precise representation, but then that representation can be compiled into other forms. For more on representation see Chapter 22, by Vickers, this volume.

In this chapter we will consider ontologies represented as categories. This will have several advantages for us, including the use of the technical tools of category theory (see Chapter 21, by Healy, this volume) to support our analysis of ontologies, the use of categorical logic (see Chapter 22, by Vickers, this volume) to make clear the link between ontologies as categories and ontologies as logical theories, and the graphical representation and reasoning of category theory to easily allow fragments of ontologies to be represented in their more usual form as graphs or trees.

As an example we present a fragment of an ontology for air traffic control systems. For those not used to category theory it will suffice to contemplate the diagram (Fig. 24.1) and we will explain the categorical constructions in that diagram in logical terms. For ease of exposition we have focused upon the part of the ontology related to aircraft planning a landing.

Nodes in the directed graph shown in Fig. 24.1 are the main subject matter of the ontology. In this case we are concerned with airports, runways, navigational aids (VOR, which stands for VHF Omni-Range, and NDB, which stands for Non-Directional Beacon), and aircraft approaches to airports. As noted by Gruber (2009), ontologies are typically represented using classes, relationships (between classes) and attributes (of instances of classes). In Fig. 24.1 classes are represented by nodes

Fig. 24.1 A fragment of an air traffic control ontology



and functional relationships by arrows. Attributes, while important and easily represented in our category theoretic approach, are suppressed to keep the diagram simple.

At this point it is worth making three remarks about our use of functional relationships.

First, all traditional (many-to-many) relationships can be represented as a “span” of functional relationships. For example Approach might be a relationship between Runway and NavAid (some navigational aids can be used as final approach fixes for particular runways). Here that is expressed by the directed edges to and faf (final approach fix). As is often the case, an attempt to “tabulate” the (many-to-many) relationship between Runway and NavAid in fact records a class that has semantic significance and which might itself be explicitly recorded as a class in other related ontologies. So, choosing to represent only functional relationships doesn’t limit our representational power, and may (in practice, does often) assist in identifying meaningful classes.

Second, because ontologies are intended to be more abstract than data models, what in many ontologies appears as a functional relationship in fact may be only a partial function. If one were to say for example that an airport has a radio frequency used for approaches, one would be largely correct. And it is certainly worth recording that an airport may have such a frequency, and that that relationship is in general functional, but strictly the function would be partial since there are uncontrolled airports. Such partiality most often arises with optional attributes and can be dealt with by introducing a (tabulated as just discussed) relation. How one does this turns out to include some theoretical surprises, and we will discuss it in more detail in Section 24.8.

Third, *is_a* relationships, among others, are denoted here by edges indicated \gg . Those edges are intended to be realised not just as functions, but as injective (one-to-one but not necessarily onto) functions. This makes sense since if every instance of X is an instance of Y , then we certainly expect that given an instance of X we have a corresponding instance of Y (ie we have a function), but furthermore no two distinct instances of X would correspond under the *is_a* relation to the same instance of Y (ie the function is injective). It is also valuable to be able

to specify that other (functional) relationships are also required to be injective. For example, the function *faf* specifies, for an approach to a particular runway, which navigational aid provides the final approach fix. We will return to the specification of \gg arrows in Section 24.3.

Remembering that Fig. 24.1 is a representation of part of a category, there are other constraints. For example, in a category, arrows can be composed and so diagrams do, or do not, commute (see again Chapter 21, by Healy, this volume). In our example, both rectangles commute. To check the commutativity of the lower rectangle we note simply that an Approach uses as final approach fix a NavAid which is located at an Airport, and that the approach is to a Runway which must be at the same airport as the navigational aid. On the other hand the triangle does not in general commute: An Approach has associated with it an Overshoot which indicates which NavAid an aircraft should go to in the case of failing to land after the particular approach. The overshoot navigational aid will not usually be the aid that was used as the final approach fix for the approach.

It is important to emphasise that whether or not a diagram commutes is a question of real-world semantics – a question that should be asked, and a question whose answer should be recorded as part of the ontology. Further examples of the importance of specifying commutative and not necessarily commutative diagrams are given in Johnson and Dampney (1994).

Finally there are other, also categorically and logically expressible, interactions between the nodes shown in Fig. 24.1. The *is_a* arrows into NavAid tell us that all instances of VOR and NDB are NavAids. There is nothing in the diagram that tells us that the “subobjects” are disjoint from each other, nor is there anything to say whether or not there can be NavAids which are of other types. If, as is the case in the real ontology, NavAid is known to be the *coproduct* of the two subobjects, then that ensures that both these conditions are satisfied. In set theoretic terms the coproduct requirement ensures that NavAid can be obtained as the disjoint union of VOR and NDB. Similarly NDBApproach can be computed as a *pullback* in set theoretic terms, which in this case is the inverse image of the inclusion of NDB’s in NavAid’s along *faf*. It selects those Approaches whose NavAid is an NDB.

Note that although NavAid and its two *is_a* arrows can be computed from other classes, it needs to be presented in Fig. 24.1 as it is the codomain of two further arrows. On the other hand the ontology would not be changed at all if NDBApproach were left out. It, and its two arrows, and the commutativity of its rectangle, are all determined and can be recomputed as needed. This flexibility is at the heart of ontology interoperation discussed in Section 24.5.

24.3 Presenting Ontologies

In the last section we noted (by example) that an ontology may seem to take a different form merely because classes may not be included although they may still be fully determined by other classes and relationships which are present. This is the essence of the semantic mismatch problem which has made system

interoperability so difficult and has motivated many of the developments of ontologies. Probably the potential complications are obvious to those who have worked with independently developed but partially overlapping ontologies: each of the ontologies contains enough information to model the application domain, but they seem hardly comparable because classes, often sharing the same name, can be two different specialisations of a more general class that doesn't appear in either ontology, and because each ontology seems to contain important classes that are absent from the other.

In this section we briefly review mathematical presentations and the forms of category theoretic presentation that the authors have found empirically sufficient for ontology engineering. See also the discussion of ontological commitment in [Chapter 22](#), by Vickers, this volume.

A *presentation* is a specification of the generators and axioms that are required to determine a mathematical object. Common examples include in group theory presentations of a group, or in the theory of formal languages presentations of a language. A presentation is important because it gives a precise and usually finite specification which suffices to fully determine the object. Yet presentations are not usually the subject of mathematical study since a given mathematical object will usually have many different presentations.

Categories are presented using *sketches* (Barr and Wells, 1995). At its most basic, a sketch consists of a directed graph (like Fig. 24.1) together with a set of diagrams in the graph (pairs of paths of arrows with common start and endpoint) which are intended to be the commutative diagrams. The category presented by the sketch is simply the smallest category, generated by the graph, subject to the commutativity of the diagrams (and hence of all others that follow logically from those diagrams).

More generally a *mixed sketch* is a sketch together with sets of cones and cocones (see [Chapter 21](#), by Healy, this volume) in the graph which are intended to be limit and colimit cones. The category presented by the (mixed) sketch is simply the smallest category with finite limits and finite colimits, generated by the graph, subject to the commutativity of the diagrams and to the requirement that the cones and cocones do indeed form limit cones and cocones.

The formal development of the theory of (mixed) sketches is not important for us here but one empirical observation is important: over a wide range of practical studies finite limits and finite coproducts have sufficed to specify ontologies. So, we will limit our attention to these kinds of mixed sketches. Furthermore, it turns out that finite limits and finite coproducts are sufficient to support, via categorical logic (see [Chapter 22](#), by Vickers, this volume) a large fragment of first order logic. This fragment is sufficient to model all usual “queries” for example. It is a powerful tool for specifications, and easily supports, for example, the specification of monic arrows which are modelled by injective functions. Indeed, an arrow is monic exactly if its pullback along itself can be obtained with equal projections.

Having settled on the presentations we use, we now note in the strongest of terms: An ontology is not its presentation. If you accept our claim that finite limits and finite coproducts should be used in the category theoretic definition of an ontology, then whenever you ask to see an ontology you will be shown a presentation. This is

simply because the category with finite limits and finite coproducts specified by the presentation will in general be infinite, so can't reasonably be shown to you. But, as with other mathematical presentations, the object of study is not the presentation, but rather the (infinite) category.

When one takes this point of view many of the difficulties of independently developed but overlapping ontologies disappear. If two such ontologies really do capture the same real-world domain then the independent *presentations* have the incompatibilities discussed at the beginning of this section, but the ontologies that they present will be the same.

Of course, other ontologists have recognised this point in their own frameworks. In particular, taking an ontology as a first order logical *theory* (rather than a collection of logical sentences) corresponds to taking the ontology as a category rather than as a presentation. Nevertheless, it is apt to be forgotten when working with tools that support and indeed display only finite fragments.

24.4 Views Versus Sub-Ontologies

An immediate benefit of recognising ontologies as categories rather than presentations comes with the definition of views. In many practical treatments a view of an ontology is in fact a sub-presentation. When such a view exists, it will behave well. Certainly a subpresentation is a view. But there are many views of ontologies whose basic classes include some which do not occur in the presentation.

Instead, a view of an ontology \mathcal{O} should be a presentation \mathbb{V} together with a mapping of that presentation into the entire ontology \mathcal{O} , not just into the presentation for the ontology \mathcal{O} .

The idea here is worth emphasising: A view of an ontology should be given just like any other ontology would be, via a presentation \mathbb{V} , but when it comes to instances the ontology generated by that presentation should be populated by instances determined by the mapping of the presentation into the ontology \mathcal{O} .

Incidentally, in the category theoretic treatment, a mapping of a presentation \mathbb{V} into an ontology \mathcal{O} is the same as a functor, preserving limits and coproducts, between the ontology \mathcal{V} generated by the presentation \mathbb{V} and the ontology \mathcal{O} . If in a particular application the ontology \mathcal{O} has sets of instances associated with its classes, then the view is obtained for any class V by following the functor to \mathcal{O} , obtaining a class in \mathcal{O} , and then seeing what set of instances is associated with that class.

We choose to work always with views and to eschew the use of subontology except for the very simple cases which arise as a result of subpresentations.

24.5 Interoperations

The last section dealt carefully with views and subpresentations because some particularly forward looking work in the 1990s was intended to develop interoperations for ontologies via an *algebra of ontologies* (Mittra and Wiederhold, 2004). The

algebraic operations were based on the “intersection” of ontologies, and we claim that the notion of *view* introduced in the preceding section supports an appropriate generalisation of the notion of intersection.

The idea is as follows.

First, let’s review the familiar notion of intersection. Given two structures A and B their *intersection* is the largest structure C which appears as a substructure of both A and B . Diagrammatically $A \leftarrow C \rightarrow B$, where the arrows are inclusions and C is the largest structure that can be put in such a diagram. Thus if we seek the intersection of two presentations of two ontologies the construction is fairly clear: We merely take the set-theoretic intersection of the classes of the ontologies, and the set theoretic intersection of the relations between those classes.

Of course, the foregoing is too naive. Two different ontologies may have two common classes X and Y , and in each ontology there may be a functional relation $X \rightarrow Y$, but the two relations may be semantically different. Perhaps we could require that the relations carry the same name when we seek their intersection, but this brings us back to the very problem ontologies are intended to control – people use names inconsistently.

Instead, we take seriously the diagram $A \leftarrow C \rightarrow B$, developed with insightful intervention to encode the common parts of A and B and record that encoding via the two maps. Thus an $f : X \rightarrow Y$ will only appear in C when there are corresponding semantically equivalent functions in A and B , whatever they might be called.

The requirement to check for semantic equivalence is unfortunate but unavoidable. Mathematically structures can be linked by mappings provided the mathematical form that they take does not contradict the existence of such a mapping, but whether such mappings are meaningful is a semantic question that requires domain knowledge.

Now we need to note that so far we have only dealt with presentations. If we want an appropriate generalisation of intersection for ontologies we need to recognise that two ontologies can have common classes. The commonality only becomes apparent when one moves from the presentations, to the ontologies. For example, two ontologies might both have a class called *Product*, but if the ontologies were developed for different domains those classes are very unlikely to be semantically equivalent. Nevertheless, it might happen that the ontologies do both deal with products of a certain type. To find an “intersection” it will be necessary to specialise both of the *Product* classes, perhaps by restricting them to products with certain attributes (likely different in the two different ontologies) or that have certain relationships with other classes (again likely different in the two different ontologies). In our experimental work such situations arise frequently, but with the limits and coproducts that are available in the two ontologies (represented as categories with finite limits and finite coproducts) we can analyse the specialisations and determine the corresponding classes in the two ontologies. Thus, if the two ontologies are called \mathcal{O} and \mathcal{O}' we develop a view \mathbb{V} together with mappings into \mathcal{O} and \mathcal{O}' obtaining $\mathcal{O} \leftarrow \mathbb{V} \rightarrow \mathcal{O}'$.

The largest such \mathbb{V} might be viewed as the “generalised intersection” of \mathcal{O} and \mathcal{O}' .

In fact, in our empirical work we can rarely be confident that we have obtained the largest such \mathbb{V} . No matter, \mathbb{V} together with its mappings provides an explicit specification of identified common classes in the two ontologies that should be kept synchronised if we expect the ontologies to interoperate. In practical circumstances two industries will have specific intentions about how they will interoperate, and so there is guidance as to which general areas of their ontologies will need to be synchronised. There may be other commonalities (for example, Person might be semantically equivalent in the two ontologies), but if as industries they keep those parts of their operations distinct we need not necessarily support interoperations on those commonalities.

It is worth emphasising here that views are better than subpresentations, and ontologies as categories with finite limits and finite coproducts are better than mere presentations of ontologies, because, at least in our experience, only in very fortuitous circumstances will ontologies be able to be “linked” via classes which happen to appear already in their presentations.

24.6 Solving View Updates

Although this chapter has dealt mostly with ontologies, rather than with the applications of ontologies (often databases of some kind) that store the instances of the classes that occur in the ontology, it is important to consider carefully the interaction between views and instances.

In one direction the interaction is straightforward. As noted in the previous section, a view’s classes should be populated with the instances from the corresponding classes in the ontology. In category theoretic terms, the assignment of instances to classes is a functor from the ontology (viewed as a category) to a category of (usually finite) sets. Then a view, being not simply a presentation \mathbb{V} but also a functor between the ontology \mathcal{V} determined by that presentation, and the fundamental ontology \mathcal{O} , can be composed with the instances functor $\mathcal{O} \rightarrow \mathbf{set}$ to yield an instances functor $\mathcal{V} \rightarrow \mathbf{set}$.

Thus modifying the recorded instances of an ontology automatically modifies the instances of any view of that ontology.

The reverse direction presents significantly more complications. In particular, if one were to treat a view and its instances as if it were simply an ontology, and one were to modify (update) the recorded instances of the view, it’s a very significant problem to determine how best to correspondingly modify the recorded instances of the ontology. In database theory this has been known as “the view update problem” and has been the subject of research for nearly 30 years.

The difficulty of the view update problem is easily underestimated when one concentrates on the classes, as we so often do when working with ontologies, and neglects the relations between classes. But fundamentally the information about an instance of a class is contained in the way that that instance is related to other instances of other classes, or indeed to attributes. Thus, when one introduces or

deletes an instance from a class in the view, an instance must be likewise introduced or deleted from the corresponding class in the ontology. But what happens to the relationships that that instance might participate in? In some cases the relationships will also be present in the view, and then we can see what we should do to them in the ontology, but inevitably there will be relationships which are not present in the view but which need to be modified in the ontology. How should we modify them if we want to update the instances of the ontology to match the instances of the view?

Over the years there have been many solutions proposed for the view update problem. We won't review them here, but we will point out that there is one solution suggested naturally by the category theoretic approach – category theory makes extensive use of so-called *universal properties* (properties like those used to define limits and colimits in [Chapter 21](#), by Healy, this volume), and there is a natural such property, well-known in category theory, which can provide arguably optimal solutions to view update problems. The required property is the existence of cartesian and op-cartesian arrows, and the details are presented in the authors' Johnson and Rosebrugh (2007).

Rather than embarking on the category theoretic details, we will proceed now to show how solutions to view update problems, when they exist, can be used to develop ontology interoperation, and hence ultimately to aid ontology engineering in the sense discussed in the introduction – engineering new ontologies by carefully developing interoperations among existing ontologies.

24.7 Interoperations with Instances

For many purposes, including an algebra of ontologies, and for many ontologists, the identification of a common view in the manner of [Section 24.5](#) suffices. A new ontology incorporating both extant ontologies and respecting their common views can be calculated by taking the colimit of $\mathcal{O} \leftarrow \mathcal{V} \rightarrow \mathcal{O}'$ in the category of categories with finite limits and finite colimits. This corresponds to Healy's "blending of theories". Nevertheless, we can ask for more.

Many ontology projects, including both BizDex and aspects of e2esu (End-to-End Service Utility), seek to develop interoperations based on independently developed domain-based ontologies. Interoperations in these cases mean interoperations at the instance level, and as we saw in the preceding section, interoperations at the instance level are more subtle than the colimit ontology would suggest.

We say that ontologies \mathcal{O} and \mathcal{O}' interoperate via a view \mathbb{V} when in the diagram $\mathcal{O} \leftarrow \mathbb{V} \rightarrow \mathcal{O}'$ the view update problems have both been solved. In such a case the ontologies support interoperating systems. Suppose that we have two information systems I and I' based respectively in the ontologies \mathcal{O} and \mathcal{O}' , then the systems can interoperate as follows. Suppose that a change is made to the information stored in system I . Immediately that results in a change to the \mathbb{V} view of I . Because we

have a solution to the view update problem for the \mathbb{V} view of I' that change in the view can be “propagated” to the information system I' . Similarly, changes in the information stored in I' are propagated via the common view to I . The result is that the two systems remain synchronised on their common views while they operate independently and, apart from the view update code, without any modification to the original stand-alone information systems.

A detailed example for e-commerce systems is presented in Johnson (2007). That paper also points out that this “full-duplex” interoperability is in fact stronger than is often required in business. Instead, a “half-duplex” approach to interoperability often suffices. This takes advantage of the empirical observation that for particular interoperating businesses information only flows one way on certain classes, and the other way on other classes, and these limited flows reduce the difficulty of solving the view update problems.

We reserve the term *interoperating ontologies* for cases like those discussed in this section where view update problems have been solved, at least in the half-duplex sense, and so the \mathbb{V} -linking of ontologies yields a corresponding and effective linking of extant systems constructed using those ontologies.

24.8 Nulls and Partial Functions

It is notoriously difficult to precisely distinguish ontologies from data models. Even Gruber’s definition (Gruber, 2009), in common with other treatments, can only broadly distinguish ontologies by discussing their expected independence of data-structures and implementation strategies. He also notes that primary keys are the kind of thing that one would *not* expect to find in an ontology.

One only half-facetious proposal put forward by the authors is that ontologies are, at least often in practice, those data models in which all functional relationships are partial. This does correspond with remarks about primary keys, since a defining property of primary keys is that they are mandatory attributes. But more than this, ontologies frequently provide a controlled vocabulary that indicates what *might* be said about instances. Thus attributes are optional, or in other words, attributes are always allowed to take null-values.

This suggests that in a theory of ontologies it is very important to determine how one represents null-values or partial functions.

There are two widely used representations, both in databases and in category theory. In one, a partial function is modelled by including an explicit null-value. Thus a partial function $X \rightarrow A$ is represented by total function $X \rightarrow A + 1$ with the same domain X , but whose codomain is augmented by summing it with a single null value. When the function is undefined on a particular $x \in X$ the value at x for the total function is by definition the extra or null value of the codomain $A + 1$. In the other approach a partial function $X \rightarrow A$ is modelled by a relation $X \ll X' \rightarrow A$ in which X' should be viewed as the subobject of X for which the function is defined. Surprisingly a careful analysis of these two approaches shows that in the

context of instances, viewed as categories of models, they are not equivalent. See the discussion of ontological problems with “not” discussed in [Chapter 22](#), by Vickers, this volume.

Although the explicit use of null-values is probably the most widespread representation of partial functions in databases, the authors choose to represent partial functions using the relation approach. Using this approach we accept ontologies in which explicit functional relations are drawn as arrows, but interpret those arrows $X \rightarrow A$ as an abbreviation standing for $X \ll X' \rightarrow A$. This is in contrast to our treatment of data models in which an arrow $X \rightarrow A$ is interpreted as a total function. When it comes to making precise category theoretic calculations in ontologies the spans $X \ll X' \rightarrow A$ need to be explicitly included.

24.9 Universal Nulls

One of the great advantages of interpreting functions in ontologies as partial functions is that view update problems much more often have straightforward solutions. Perhaps this is easy to see. If a view includes a class, but not an attribute of that class which is present in the ontology, then solving a view update problem will be very difficult. After all, new instances of the class in the view don't come with an attribute value, but in the ontology, at least if functions are interpreted as total functions, each new instance needs to be associated with a particular value for the attribute. Recall that view update problems are solved using category theoretic universal properties. When a function needs to take a value, but no value is in any sense canonical, then there is very little chance of finding a universal solution.

One might reasonably expect that when functions are allowed to be partial, the null-value will in some sense be canonical. Certainly, assigning the null-value in cases where no other value is determined by the view is the minimal change. Unfortunately, an explicit null-value is not category theoretically distinguishable from any other value of an attribute, and so the difficulty in finding a universal solution remains.

Happily, using the relation approach from the previous section, assigning a null-value to an instance x in the relation $X \ll X' \rightarrow A$ simply means not including x in the subobject of defined values X' . This is again the minimal change, but this time it is also minimal with respect to natural transformations among the set-valued functors which keep track of the assignment of instances to classes. It turns out that this minimality is exactly what is required to provide a universal solution for the view update problem.

24.10 Conclusion

Developing interoperating systems is one of the most important uses of ontologies. Over recent years category theory has been used to develop new approaches to ontology presentation, and new solutions to view update problems. View update solutions can be used to engineer systems interoperation.

In this chapter we have studied the relationship between category theoretic specification and ontologies. We have noted how views can be used in a general way to support calculating the colimit of two ontologies so as to create a new ontology which includes them both and recognises their commonalities. We have shown further how solving view update problems can lead to systems interoperation without a need to modify the basic systems. In this latter case we call the ontologies *interoperating ontologies*. We have noted briefly how view update solutions can be aided by the limitations of half-duplex interoperation, and by the very common if often inexplicit use of partial functions in ontologies.

Over all we have found that new and mathematically precise treatments of difficult problems in the foundation of ontologies and information systems support ontology interoperation.

It's always pleasing when theoretical developments yield practical advantages and new insights as well as stronger theory.

References

- Barr, M., and C. Wells. 1995. *Category theory for computing science*, 2nd edition. Menlo Park, CA: Prentice-Hall
- Bontas, E.P., and T. Christoph. 2006. Ontology engineering: A reality check. In *The 5th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE2006)*, LNCS 4275, eds. R. Meersman, and Z. Tari et al., 836–854, Springer.
- BizDex is an e-Business framework that incorporates common standards. <http://www.standards.org.au/cat.asp?catid=37&contentid=73> Accessed 2 Jan 2008.
- End-to-End Service Utility (E2ESU). <http://www.e2esu.eu/public/>. Accessed 19 June 2010.
- T. Gruber. 2009. Ontology. In *The encyclopedia of database systems*. eds. L. Liu, and M.T. Özsu, Berlin: Springer.
- M. Johnson. 2007. Enterprise software with half-duplex interoperations. In *Enterprise interoperability: New challenges and approaches*. eds. G. Doumeingts, J.P. Mueller, G. Morel, and B. Vallespir, 521–530. Springer, Berlin.
- Johnson M., and C.N.G. Dampney. 1994. On the value of commutative diagrams in information modelling. *Springer Workshops in Computing*, eds. Nivat et al., 47–60. London: Springer.
- Johnson, M., and C.N.G. Dampney. 2001. On category theory as a (meta) ontology for information systems research. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS2001)*, 59–69. ACM Press.
- Johnson, M., and R. Rosebrugh. 2007. Fibrations and universal view updatability. *Theoretical Computer Science* 388:109–129.
- Mitra, P., and G. Wiederhold. 2004. An ontology – composition algebra. eds. S. Staab, R. Studer. *Handbook on ontologies*, 93–113. Heidelberg: Springer International Handbooks on Information Systems.
- Smith, B., and C. Welty. 2001. FOIS introduction: Ontology – Towards a new synthesis. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS2001)*, 3–9. ACM Press.