

Alexander Ollongren

Astrolinguistics

Design of a Linguistic System for
Interstellar Communication Based on
Logic

 Springer

Astrolinguistics

Alexander Ollongren

Astrolinguistics

Design of a Linguistic System for
Interstellar Communication Based on Logic



Springer

Alexander Ollongren
Advanced Computer Science
Leiden University
Leiden
The Netherlands

ISBN 978-1-4614-5467-0 ISBN 978-1-4614-5468-7 (eBook)
DOI 10.1007/978-1-4614-5468-7
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012945935

© Springer Science+Business Media New York 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

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

*To the memory of Prof. Dr. Hans Freudenthal
and to Gunvor Ollongren*

Preface

In my perception the author of the present book appeared all of a sudden “from elsewhere” on the scene of academics and others concerned with various projects in the field of the search for extraterrestrial intelligence (SETI). This was in 1998. Dr. Ollongren’s entrance in the field was due to a change in his research interests. They had apparently shifted from dynamical astronomy and theoretical computer science to the topic of the art and science of communication with extraterrestrial intelligence (CETI)—closely related to my work at the SETI Institute. His approach was and is, however, completely different from mine where human psychology is always prominently present in the background. This emeritus professor, astronomer, and theoretician whom I got to know, had a keen interest and experience in mathematics and logic. He seemed, though hesitatingly, prepared to consider formally modeling psychological aspects of human behavior in connection with interstellar message construction for possible contacts with alien highly developed intelligent societies. I, on the other hand, recognized immediately that Dr. Ollongren’s approach for the design of a *Lingua Cosmica* was fundamentally new, rested on a solid and sound base in logic, and carried great potential for applications. His LINCOS has only very little in common with the interesting language for cosmic intercourse of the same name introduced in 1960 by his fellow countryman and colleague Professor Hans Freudenthal.

The present book describes the new LINCOS, also at work. This *lingua* with a simple formal language at the base is actually a system because of the multilevel structure as explained in the book. Certain aspects of the system are in my view most remarkable. The fact that LINCOS assertions can (and need to) be verified within LINCOS itself means that the system is strongly self-contained—a property of natural languages, but unusual for formal systems. The implication of this property (in fact considered to be a requirement in the design) is that formal assertions are always correct. This aspect in its turn has important consequences for the problem of decoding and interpretation. An alien receiver of a message coded in Ollongren’s LINCOS can in principle apply a universal semantic engine for unraveling structural properties of the system. Furthermore there is the prominent absence of logical reasoning according to *tertium non datur* (the rejection of the law of the

excluded middle)—not one of the first concerns in my research in psychological aspects of human reasoning figuring in the topic of construction of interstellar messages. The overall most important aspect and viewpoint of the material in this book is, however, that the linguistic system designed and employed is constructive and deterministic. An extension of the typing system employed to weak typing (involving consequently non-determinism) and symbolic computing is innovative. Ways and means (with roots in intuitionism) for attaining all of this are explained in some detail. The erected framework with a beauty of its own is most useful for applications in and development of CETI.

In my opinion efforts in the field of construction of messages for ETI should also take into consideration possibilities of symbolically coding human behavior. Fortunately LINCOS has in the realm of declarative (i.e. introductive) sentences a practically unlimited capacity of describing *static* aspects of this kind. This is for example illustrated in Dr. Ollongren's paper with a treatment of human morality in my as yet unpublished book. In addition there is also a potential for describing *dynamic* interactions between humans. Examples of these are provided in the present book. Humans exchanging information is one case studied and explained. Since human behavior often (indeed not always) exhibits logic structure, the new *Lingua Cosmica* provides a powerful apparatus for formal analysis of some classes of behavioral actions. Products of such analyses can in turn be sublimated in messages for ETI.

I was very interested to learn that Springer has agreed to publish the present book as the relatively new discipline called astrolinguistics by the author is gaining importance all the time due to advances in observational astrophysics, astrochemistry, and astrobiology, but of course also cosmology. Building stones for life as we know it exist in interstellar space. It is no exaggeration to state today that our galaxy apparently teems with planets; there is ample observational evidence of that. On at least one habitable planet the symbolic species living there is concerned with developing methods and means enabling them to contact other completely unknown symbolic intelligent species. The views taken and the material presented in the present book can play a prominent role in the nontrivial efforts on Earth in this field. I certainly hope that the book will attract a wide circle of readers and will help to advance future work on the construction of interstellar messages. I wish the book a bright future.

Professor Douglas A. Vakoch PhD, SETI Institute and California Institute of Integral Studies, Mountain View, California, United States

Contents

Part I Calculus of Constructions

1	Types and Declarations	3
	Intention	3
	Typed Entities	3
	Functions.....	5
	References.....	7
2	Functions and Induction.....	9
	Intention	9
	Constructing Entities.....	9
	Inductive Entities	11
	Facts	14
	Connectives, Continued	16
	Typing Existence and Equality	17
	Booleans.....	19
3	Hypotheses	23
	Intention	23
	Contradictions	23
	Double negation	25
	Hypotheses	28
	Reference	28
4	Higher Orders and Inductive Structures.....	29
	Intention	29
	The Combinators.....	30
	Bounded Matrojshka.....	32
	Syntactic Structures	35
	Reference	38

Part II Facts

5 Simple Facts	41
Intention	41
Elementary Facts.....	41
Non-elementary Facts, Elimination Applied	43
Annotation 1.....	44
Annotation 2.....	45
Existence Revisited	45
6 Compounds	47
Intention	47
Commutativity, Transitivity and Distributivity	47
Modus Tollens.....	48
Logic in Sentences	48
Complexity of the Stage.....	52
7 Aristotelian Theatre	53
Intention, Logic of Sentences	53
Simple Cases.....	54
Aristotelian Conversions	56
Logical Implications	57
Figures.....	60
Reference	63
8 Wittgenstein's Theatre	65
Intention	65
Introduction.....	65
Verification Machinery.....	67
Setting the Stage	68
Simple Facts Verified	69
Less Simple Facts Verified.....	70
Generalisation, More Advanced Verifications	73
Notes on Computer Implementation.....	74

Part III Annotations in LINCOS

9 Logic Contents of Texts	77
Intention	77
Considerations.....	77
An Ancient Text	78
Message Content.....	78
Simonides' Definition of Justice	80
Thrasymachos' Definition of Justice	80
Socrates' Definition of Justice	82
Discussion	84
References.....	84

10	An Astrolinguistic Experiment	85
	Intention	85
	About the Second Level	85
	The Experiment.....	86
	Interpretation.....	89
	The Experience Learns	90
	Appendix	91
	References	91
11	Aspects of Truth	93
	Intention	93
	Teaching Truth	93
	Verification of Truth.....	95
	Enriching the Environment	97
	Static Relations	98
	Super- and Subvenience.....	99
12	Human Altruism	101
	Intention	101
	Introduction.....	101
	Altruism	103
	Moralism	103
	Types of Moral Behaviour	104
	Duty and Obligation.....	105
	Summary and Conclusion	105
	References.....	108

Part IV Interpretation of LINCOS

13	Self-interpretation	111
	Caption	111
	Intention	111
	Disposing of ELIM	112
	Conjunction.....	113
	Disjunction.....	114
	Inductive Self-interpretation	115
	References.....	116
14	Pictorial Representations	117
	Intention	117
	Pioneer Plaque	117
	A Course in Latin.....	118
	Individuals.....	119
	Intentions of Subjects.....	121
	Reference	121

15 Representation of Music	123
Intention	123
Considerations.....	123
Musical Units in Gamelan	124
Gamelan Performance.....	126
Explaining LINCOS.....	128
Notes	129
References.....	130
16 Signature of LINCOS	131
Intention	131
Introduction.....	131
Subjects and Predications Revisited	132
Implication and Maps	132
Subjects and Predications, an Example	134
λ and the Signature	134
Note.....	135
Reference	135
 Part V Processes in LINCOS	
17 Representing Processes	139
Intention	139
Introduction.....	139
Sequences.....	140
Channels.....	141
Example: Production of Strings of Symbols.....	142
State Vectors.....	142
Example: A Producer/Consumer System.....	143
Example: A Producer/Client System	144
Note.....	145
18 Cooperating Sequential Processes	147
Intention	147
Introduction.....	147
Concurrency	148
Arbitration.....	148
Five Dining Philosophers.....	149
Conclusion	150
References.....	151
19 Hamlet in LINCOS	153
Intention	153
Introduction and Basics.....	153
Parallel Processes.....	154
The Arbiter for Parallel Processes	155

Application, Opening Act in Hamlet	156
Conclusion	158
References	159

Part VI Symbolic Computation

20 Basics	163
Intention	164
Introduction	164
LINCOS Enriched	165
Metric Two-Dimensional Space	167
Reference	168
21 Relativistic Particle Motion	169
Intention	169
Non-relativistic Space	169
Special Relativity Theory in LINCOS ⁺	171
General Relativity (GRT) in LINCOS ⁺	172
Conclusion	174
Reference	175
22 Two-Body Motion	177
Intention	177
Co-ordinates	177
Orbital Elements	178
Euclidean Metric	179
Conclusion	179
References	179

Part VII (Un)Certainty

23 Certain Existence	183
Intention	183
Introduction	183
Propositions Exist	184
<i>All</i> and Lambda Binding	184
<i>All</i> and <i>Some</i> in Existence Maps	185
Procreation	186
Conclusion	188
Reference	188
24 The Uncertain Alien	189
Intention	189
Background	189
Peirce's Law Uncertain	190
Uncertainty Risks	191

Uncertainty Instead of Falsity	192
An Open Question for ETI.....	193
Notes	194
Reference	194
 Appendix A: Declaration of Principles Concerning the Conduct of the Search for Extraterrestrial Intelligence.....	 195
 Appendix B: Preliminaries.....	 197
 Appendix C: History	 207
 Appendix D: A Gentle Introduction to Lambda and Types.....	 217
 Appendix E: Postscriptum	 227
 Appendix F: Summary in Russian	 239
 Appendix G: Curriculum Vitae of Alexander Ollongren (* 1928, Sumatra)	 241
 Index.....	 243

About the Author

Alexander Ollongren began his career as a mathematical astronomer at the Department of Astronomy in Leiden University in the Netherlands, where he obtained his PhD. He then left the university and worked almost 2 years at the Research Center of Celestial Mechanics at Yale University in the USA. Ollongren returned to Holland and became director of the newly established computer center of Leiden University. He spent a sabbatical leave as a visiting scientist at the IBM Laboratory in Vienna and was later appointed full professor of theoretical computer science at the Department of Computer Science of Leiden University. He was a guest professor in the same science for about a year at Linköping University in Sweden. After retirement he became a member of the Permanent Study Group Search for ExtraTerrestrial Intelligence (PSGSETI) of the International Astronautical Academy. He has written books and scientific articles on the semantics of programming languages.

Part I

Calculus of Constructions

Introduction

The point of view taken in this treatise is that a *lingua cosmica* for interstellar message construction should have *multilevel* structure—the new LINCOS as described in this book is therefore a linguistic *system*, albeit in a restricted sense. A natural language is also a system in which one can distinguish aspects as syntax and semantics—but also pronunciation and representation of phonemes, words, and sentences in writing, all of these strongly interconnected. In LINCOS the situation is different: the levels are rather loosely connected. In the simplest case one is concerned with two levels: a message consists at one level of a text in some natural language supplemented, at another (*meta*) level, with descriptive *annotations* in a formal system, in our case LINCOS. A third level may be employed to give additional information, on the text or concerning the annotations. In general the annotations do not pretend to give complete descriptions of the (essentially logic) contents of the text concerned. At all levels *linear notation* is employed—this is evidently important when considering actual transmission of messages in interstellar space. We are mainly concerned with annotations at the secondary level and we keep strictly linear notation in the book as well.

These design criteria raise the question whether LINCOS itself can (or should) explain the formalities involved at the various levels. Will receivers of an interstellar message written in LINCOS understand that two (or more) levels are involved? The latter matter can be resolved by clearly separating the levels, e.g., by using a kind of characteristic delimiters. In the author's view there is no need to explain in a linguistic system for interstellar message construction the formalities (i.e. the grammar) of the natural language used in the textual level; this would be not completely possible anyway. At the same time care should be taken to explain the conventions of the formal system used at a secondary level. The present book pays indeed attention to some possibilities (perhaps by using a third level as well), in particular of doing so within LINCOS itself. Helpful in this respect is the fact that the syntax of the formal calculus employed is rather simple. In the present part, however, we do not discuss this issue in detail.

Instead we explain in this part of the book the background of the formal system, embodied in LINCOS as employed at the secondary level. The formal system is based on the Calculus of Constructions (CC), the foundation of which is the theory of types. CCI, i.e., CC extended with *induction*, is used in argumentation: starting from basic assumptions and using deductive and inductive rules conclusions are achieved guaranteed to be logically correct. This important aspect is inherited in the LINCOS system. Since both CCI and the theory of types contain unusual concepts and perhaps unfamiliar constructions, it is appropriate to give the reader the opportunity to familiarize him/herself with these relatively new theories. Outside of the chapters in Parts I–VII, the Appendix D of the book consists of a separate chapter in which the necessary basic information on type theory is supplied. This is done in the form of a “gentle introduction” to the λ *Calculus* and *type theory*, the conceptual frameworks and interrelations between them.

The purpose of Chapter 1 of the book is to help readers getting to know how the theories are put to work. This is effectuated by means of rather simple applications. More advanced concepts as induction and constructions and their uses are explained in Chapter 2. These two chapters set the scene as far as notation is concerned to be used in annotations at the secondary level written as LINCOS terms. Chapters 3 and 4 discuss more advanced applications.

Chapter 1

Types and Declarations

Let us define as type A: all White Rabbit's friends and family members.

from: G. Kamsteeg, PhD dissertation (Kamsteeg 2001), inspired by: Lewis Carroll, Alice in Wonderland

Intention

The purpose of the present chapter is to introduce elements of the *Lingua Cosmica* proposed and discussed in this book, in an informal and gentle way. Constructs employed in the language are explained by numerous examples. Various ways of introducing entities are illustrated, distinctions between them are described—so that the reader may familiarize him/herself with the employed mechanisms without having to grind through grammatical formalities. Note, however, that the grammatical base for building expressions in LINCOS is extremely simple as shown in the Appendix D of the present treatise. That gentle introduction to the λ Calculus and *type theory*—theories in mathematics and logic—describes the basics of the untyped as well as the typed λ Calculus. The typed λ Calculus and the so-called Calculus of Constructions are the pillars of the new LINCOS.

Typed Entities

In using a Calculus of Constructions (CC) in message construction in an astrolinguistic context (that is to say in annotating messages for interstellar communication or supplying commentaries to them, a central idea in LINCOS) one is concerned with certain expressions, also called *terms*, and a notational formalism. The CC used here

is based on the notion of *types*, themselves left un-interpreted. All basic entities and expressions built from them have types. Constants and variables are basic entities—but expressions are also considered to be entities. An important rule is that each entity has *one and only one type*. Types, in order to be recognizable as such, are written in a separate font. There are three basic constants, Type, Set, and Prop. Their types are

Prop : Type. Set : Type. Type : Type.

These are available and need not be introduced explicitly. The remarkable situation that the type of Type is Type, is discussed at the end of this section. *Declarations* are the means of introducing (new) entities explicitly. Note that above expressions are in accordance with the rule “If an entity *a* has type *A* we write *a* : *A*.” But this is not a valid declaration of *a*. In writing *a* : *A* it is supposed that *A* is either itself in some sense Type, Set, or Prop, or has been explicitly declared. *A* could have been declared by, e.g.,

CONSTANT A : Type.

This is an example of an (explicit) *type declaration* of a constant. Note that we could also have declared *A* to be a variable by VARIABLE A : Type. Whichever to choose depends on the intended use of the entity. CONSTANT and VARIABLE are declarators.

An example of a way to introduce a new entity *a* is one and only one of the declarations

CONSTANT a : A. VARIABLE a : A.

Note: for the other two basic constants we could have declared

CONSTANTS Set, Prop : Type.

Set is introduced in order to shift the burden off Type. Prop is introduced in order to be able to use the logical connectives from propositional calculus directly (see Chap. 2). The binary connectives, usually written for ease of reading by humans in the *infix notation*, are

/\ (and), \/ (or), → (implication),

while the unary connective is written ~ (negation, prefix notation).

Suppose *A* and *B* have been declared by

VARIABLES A, B : Prop.

Then *by convention* expressions like $A \wedge B$, $B \wedge A$, $A \vee B$, and $B \vee A$ are of type Prop (sometimes we say that they “are in Prop”), as well as laws like $A \wedge B \rightarrow A$, $A \wedge B \rightarrow B$ (projections) and these two well-known laws

$(A \wedge (A \rightarrow B)) \rightarrow B$ (Modus Ponens).
 $(\sim B \wedge (A \rightarrow B)) \rightarrow \sim A$ (Modus Tollens).

Note that these projection laws and the two deduction rules could be declared as well, but then in the form of hypotheses. But they need *verifications* before that can be done. Since CC as formulated here is based on the type theory of constructions

and has been somewhat influenced by early versions of the French Coq implementation of it (the *Proof Assistant*) (Coq Development Team 2010), it should be no surprise that these projections and rules can in fact be verified internally in this particular version of CC. For that purpose, however, a rather general framework is necessary. In order to arrive at that stage eventually, we shall need various additional concepts, amongst others that of a *function*. An introduction to this topic is the following.

Let the entity X be declared as a constant or variable of type Set . Since X is of type Set , one can also say that X *resides* (or *lives*) in Set , notation $X : \text{Set}$. X may have residents too—if $x : X$, then x resides in X . Again: the entity x may have residents too and thus a hierarchy of types results. We mentioned already the important rule that any variable or constant, in fact any type expression, resides in one and only one type. On the other hand a type can have many residents. The hierarchy of types is a *partial order* with the (unique) Type at the base. Because of that the partial ordering is well-founded.

An important note is the following: Set and Prop reside in Type . Type cannot reside in Set or Prop because otherwise one of these would have been given a resident a priori—without declaration. So we have the remarkable situation that (by convention) Type resides in *itself*. In addition, there is the rule that except Type no type can reside in itself. The reason for that is: suppose $Y : Y$, where Y is not Type . That means that Y must have been explicitly declared, say of type Set . But then Y resides in two types, which is forbidden.

Functions

The logical connectives have types too. In order to explain them the symbol $\rightarrow$ is given a second meaning (so it becomes overloaded). What interpretation of $\rightarrow$ is meant, is always clear from the context in which the symbol appears. Let A and B be declared. Then declare

$$\text{CONSTANT } f : A \rightarrow B.$$

Here f can be considered to be a fixed function (or *mapping*) of arity 1 which takes residents from A to residents from B (one to one), i.e., f can be *applied* to a resident of A . Let $a : A$, then $(f\ a) : B$, using prefix notation for functional application. Note that $(f\ a\ b)$ is always undefined, even if $b : B$. In addition note that no computation is involved: this is because the image (say b) under the mapping is not yielded. In much the same way the connective $\wedge$ is considered to be a function (of arity 2). It could be declared as

$$\text{CONSTANT } /\ : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}.$$

Which means that if $P, Q : \text{Prop}$, then $(\wedge P) : \text{Prop} \rightarrow \text{Prop}$, and $(\wedge P\ Q) : \text{Prop}$, as expected. By convention we write usually for the last expression $(P \wedge Q)$ instead, using infix notation. The other connectives could be declared as functions like this

```

CONSTANTS /\, → : Prop → Prop → Prop.
CONSTANT ~ : Prop → Prop.

```

This looks perhaps rather strange, but that is because $\rightarrow$ is overloaded (as mentioned above). Expressions using the logical connectives $\vee$ and $\rightarrow$ are also usually written in infix notation.

In addition to the declaration of constants and variables, we can declare the above-mentioned projection laws and the two deduction rules Modus Ponens and Modus Tollens as hypotheses using the delimiter `HYPOTHESIS`. Evidently a law to be introduced as a hypothesis must be correct, preferably verified by internal means (i.e., constructively within CC). We shall see later how correctness can be guaranteed a priori. Let the variables A and B have been declared as type `Prop`. The projection laws can be declared as follows

```

HYPOTHESIS proj1 : (A /\ B) → A.
HYPOTHESIS proj2 : (A /\ B) → B.

```

Note that these two entities seen as functions have arity 1, and $(\text{proj1 } x)$ and $(\text{proj2 } x)$ are only defined if $x : (A \wedge B)$. The deduction rules can be declared as

```

HYPOTHESIS MP : (A /\ (A → B)) → B.
HYPOTHESIS MT : (~B /\ (A → B)) → ~A.

```

These laws are evidently not only valid for the specific declared A and B , but also for any objects x and y of type `Prop`. So we have the generalized version of the laws as follows:

```

HYPOTHESIS proj1-gen : (∀ x,y : Prop) (x /\ y)
→ x.
HYPOTHESIS proj2-gen : (∀ x,y : Prop) (x /\ y)
→ y.
HYPOTHESIS MP-gen : (∀ x,y : Prop) (x /\ (x →
y)) → y.
HYPOTHESIS MT-gen : (∀ x,y : Prop) (~y /\ (x →
y)) → ~x.

```

The quantifier $\forall$ (for all) need not be written in LINCOS: because we let the round brackets imply by convention that the variables are “all-quantified,” i.e., that all residents of the type indicated are considered. Consider now a few types in above expressions. Evidently

```

A, B : Prop
(A /\ B) → A : Prop

```

so that

```

(x,y : Prop) (x /\ y) → x : Prop.

```

In addition to this here are some examples of types

```

proj1 : (A /\ B) → A
(proj1-gen A B) : (A /\ B) → A

```

$$\begin{aligned}
&\text{proj2} : (A \wedge B) \rightarrow B \\
&(\text{proj2-gen } A \ B) : (A \wedge B) \rightarrow B \\
&(\text{MP-gen } A \ B) : (A \wedge (A \rightarrow B)) \rightarrow B \\
&(\text{MT-gen } A \ B) : (\sim B \wedge (A \rightarrow B)) \rightarrow \sim A.
\end{aligned}$$

Let the symbol “=” denote equality, i.e., the entities occurring to the left and to the right of this symbol in equations are equal (undistinguishable) as far as type is concerned (Leibniz equality over types). Since they have the same type, they can be used interchangeably. Note, however, that equality as used here is a *meta* notion, as opposed to the use of “:=” in type expressions, e.g., (inductive) definitions. Some equalities (for declared A and B) are

$$\begin{aligned}
&(\text{proj1-gen } A \ B) = \text{proj1} \\
&(\text{proj2-gen } A \ B) = \text{proj2} \\
&(\text{MP-gen } A \ B) = \text{MP} \qquad (\text{MT-gen } A \ B) = \text{MT}
\end{aligned}$$

In the sections above we have been concerned with declarations, the simplest but basic aspect of the Calculus of Constructions (CC). In addition we have met now a rather primitive way of declaring functions, using the delimiters **CONSTANT** or **HYPOTHESIS**. In this way, a function is given in essence only a name and its type. Those introduced by **CONSTANT** are un-parametrized: they can have arguments but these must be supplied externally. It is clear that we need much more sophisticated ways of introducing functions in the discourse, not by their names and types only, but including their bodies (i.e., descriptions of what their “effects” or “actions” are). That brings us to the topic of defining parametrized functions, best treated in a framework including induction, entering the realm of CCI, the Calculus of Constructions with Induction.

References

- G. Kamsteeg *Formalization of Process Algebra with Data in the Calculus of Constructions with Inductive Types* (2001), PhD Thesis Leiden University
 Coq Development Team *The Coq Proof Assistant*, Versions 6 (2000)—8.3 (2010), INRIA, Paris

Chapter 2

Functions and Induction

Intention

The “core business” of the LINCOS system in the context of astrolinguistics discussed in the present treatise, is the *construction* of entities (i.e., types). This aspect is inherited from CC, the calculus of constructions, the base of the linguistic system, as mentioned in Chap. 1. In the present chapter a review is presented of various ways of achieving such constructions. For reasons of perspicuity, we use first a simple example in real life: the static situation of a book lying on a shelf. Any other comparable situation will do equally well. Evident “real” relations for this case are expressed in the system in an understandable way. *En passant* the basic logical operators from classical propositional logic are reviewed also in the formalism of functions and induction: conjunction, disjunction, negation, and implication. In this natural way the powerful and useful concept of *induction*, in which entities are defined in terms of themselves, appears on the scene. In this way the realm of CCI, the calculus of constructions including induction, is easily entered. We conclude with reviewing existence and equality considered to be functions as well.

Constructing Entities

Declaring entities (i.e., stating their “existence” by a kind of *deus ex machina*, or, rather to say, stating that they are available, see Chap. 1), is the simplest way of introducing them, but that can hardly be called constructive. Suppose the types A and B and the function $f : A \rightarrow B$ have been declared and that A is the case, i.e., it is known that there “exists” an entity a of type A. We have seen that in this case $(f\ a) : B$, so we have here a construction of an entity of type B. This kind of construction is basic but its usefulness is evidently restricted. An improvement is achieved by defining *parametrized* functions. The following gives, by way of an example, a brief introduction the way functions are defined and put to use. Consider the sentence

“Alice in Wonderland” is a book. It lies on some shelf #1.

This sentence in itself is not particularly interesting—it is certainly not intended to be part of a message for ETI. But it is representative of situations like: some human is living in a numbered house, the house is situated in a named street, the street is part of a named town, etc. So the above sentence can serve as a template for this kind of informative expressions. Let us see what annotation to the sentence, in the sense of LINCOS, could be given. To begin with declarations, setting what is called the *environment*, are needed.

```
CONSTANTS book, shelf : Set.
CONSTANT "Alice in Wonderland" : book.
CONSTANT #1 : shelf.
CONSTANT is-book : book → Prop.
CONSTANT is-shelf : shelf → Prop.
```

Note that the verb “to be” and its declination “is” are not declared. Some other words in the sentences will also be absent in the annotation. That a book lies on a shelf is expressed by a *function definition* (introducing the short name BS for the notion “book lies on shelf”),

```
DEFINE BS : book → shelf → Prop :=
  [λ x:book; y:shelf] (is-book x) /\ (is-shelf y).
```

The lambda term figuring here, the so-called *lambda abstraction*, informs that BS requires two arguments (arity two), the first of type book, the second of type shelf. This is because

```
(x : book) (is-book x) : Prop.
(x : shelf) (is-shelf x) : Prop.
[λ x:book; y:shelf] (is-book x) : book → shelf →
Prop.
```

as required because of the type of BS. In the definition of BS, two variables, x and y, are introduced not explicitly by declaration, but implicitly—they are said to be (locally) *lambda bound*. The symbol λ can be omitted under the convention that square brackets indicate lambda abstraction. The dot between the sequence of local variables and the body of the abstraction (here (is-book x)) is omitted as well. The symbol “;” in the lambda form is used as a separator and the symbol “:=” in the definition is used to express that BS is the same as the λ term. The two types are the same, in fact both are

```
book → shelf → Prop.
```

In other words they can be interchanged (this is Leibniz equality over types). As “Alice in Wonderland” is a book (one can say that logically book is the case) and #1 is a shelf, (logically speaking shelf is the case), we have

```
(BS "Alice in Wonderland" #1) : Prop.
```

Note that one expects that

```
(BS "Alice in Wonderland" #1) →
  (is-book "Alice in Wonderland") /\
  (is-shelf #1).
```

is the case, i.e., that an entity (give it the name f.2.1) can be constructed which has this type. We write

```
FACT f.2.1 : (BS "Alice in Wonderland" #1) →
              (is-book "Alice in Wonderland")
              /\ (is-shelf #1).
```

This is *not* a declaration because facts needs verification—a term of the correct type is to be constructed (see Chap. 5). In order to do so, we need an important new concept.

Inductive Entities

We shall use a restricted form of recursive definitions (called *inductive definitions*) of functions, i.e., definitions of maps in terms of themselves under strict rules of formation and interpretation. How to achieve that is described in this section by writing inductive definitions for four cases:

The binary logical *conjunction* function $\wedge$ (renamed here by “and”)

The binary *disjunction* function $\vee$ (renamed by “or”)

The unary *negation* $\sim$ (renamed by “not”)

We have seen that the type of the first two is $\text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$ and the type of negation is $\text{Prop} \rightarrow \text{Prop}$. A special position is taken by

the *implication* function, also written as $\rightarrow$ (if necessary, renamed by “imp”), also of type $\text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$, denoting here a mapping.

The defined functions are *parametrized* by means of the lambda forms in the position immediately after the function identifier. The variables bound to types in the lambda forms are the first of a sequence of arguments of the function defined.

Note the convention that brackets in type expressions associate to the *right* (one can say that they *cluster* on the right-hand side). So, using here the meta symbol = for general equality (not just type sameness), we have

$$w \rightarrow (x \rightarrow (y \rightarrow z)) = w \rightarrow x \rightarrow y \rightarrow z.$$

In the following descriptions the concept of verifiable facts is introduced. However, facts themselves are discussed in a broader sense in Part II of this book. The present chapter is concluded with brief discussions on *existence* and *equality* functions. But first consider the following more elementary cases.

Conjunction

The introduction rule for and (the same as $\wedge$) parametrized by X and Y is

```
INDUCTIVE and [ $\lambda$  X, Y : Prop] : Prop :=
Conj : X  $\rightarrow$  Y  $\rightarrow$  (and X Y).
```

The *selector* `Conj` with the type shown serves to identify the so-called *induction hypothesis*; there is only one here, it is $X \rightarrow Y$. Note in passing that

`and` : $\text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$

as required. In addition

`Conj` : $(X, Y : \text{Prop}) \rightarrow X \rightarrow Y \rightarrow (\text{and } X \ Y)$.

For A and B of type `Prop` we have the fact

`FACT f.2.2` : $A \rightarrow B \rightarrow (\text{and } A \ B)$.

verified by `f.2.2 = (Conj A B)`. The equality means that `f.2.2` is the same as the expression `(Conj A B)` in all respects, of course also with respect to type.

Suppose that we wish to apply the above inductive definition of conjunction to verify the implication $(\text{and } A \ B) \rightarrow P$, for some A, B either declared or constructed as terms of type `Prop` and some conclusion P also of type `Prop`. The verification is achieved by constructing a resident of P . That is achieved as follows

$[\text{h} : (\text{and } A \ B)] \ (\text{ELIM } \text{h} \ [\lambda \text{ a:A; b:B}] \text{p}) :$
 $(\text{and } A \ B) \rightarrow P$

if and only if

$(\text{ELIM } \text{h} \ [\lambda \text{ a:A; b:B}] \text{p}) : P$.

In words: the induction hypotheses associated with h , the first argument of `ELIM`, are to be *eliminated*. Because $\text{h} : (\text{and } A \ B)$, the only induction hypothesis is $A \rightarrow B$, the one selected by `Conj`, substituting A for X and B for Y . Elimination of it in view of P is achieved by constructing a resident of $A \rightarrow B \rightarrow P$.

With the second argument of `ELIM`, a λ term, a and b are introduced as *ad hoc* local variables with their types (abstracted by λ). The types are A and B , because the induction hypothesis is to be satisfied. With the help of h , the local variables and possibly the selector `Conj`, p is to be constructed of type P . If that succeeds, then we have

$[\lambda \text{ a:A; b:B}] \text{p} : A \rightarrow B \rightarrow P$. Finally

$(\text{ELIM } \text{h} \ [\lambda \text{ a:A; b:B}] \text{p}) : P$

because the induction hypothesis has been eliminated. So the verification is achieved.

Elimination procedures expressed by the rather elusive function `ELIM` are of prime importance in applications. Here are a few choices for P used above to illustrate the use of `ELIM` in the construction explained.

A special case, using $*$ to indicate “don’t care” for an argument, is

$[\lambda \text{ h} : (\text{and } A \ B)] \ (\text{ELIM } \text{h} \ [\lambda * :A; * :B] \text{h}) :$
 $(\text{and } A \ B) \rightarrow (\text{and } A \ B)$.

Note that the two projection rules of Chap. 1 are verified immediately by using ELIM

```
[λ h : (and A B)] (ELIM h [λ a:A; * :B] a ) :
  (and A B) → A.
[λ h : (and A B)] (ELIM h [λ * :A; b:B] b ) :
  (and A B) → B.
```

Note further

```
[λ h : (and A B)] (ELIM h [λ a:A; b:B] (Conj B A b a) )
: (and A B) → (and B A) .
```

This is because

```
(Conj B A) : B → A → (and B A) .
(Conj B A b a) : (and B A) .
```

The inductive definition of the conjunctive connective can be used for the construction of the entity f.2.1 above. Here is the construction

```
f.2.1 =
[λ h : (BS "Alice in Wonderland" #1)]
(ELIM h [λ x : (is-book "Alice in Wonderland"); y :
(is-shelf #1)]
(Conj (is-book "Alice in Wonderland") (is-shelf #1)
x y)) .
```

The type of the above expression (and so of f.2.1, because they are equal) is $(BS \text{ "Alice in Wonderland" } \#1) \rightarrow (is\text{-}book \text{ "Alice in Wonderland"}) \wedge (is\text{-}shelf \#1)$. Therefore f.2.1 is a resident of that type, verifying the fact.

Disjunction

The introduction rule for or (the same as $\vee$) parametrized by X and Y is

```
INDUCTIVE or[λ X, Y : Prop] : Prop :=
  Prim : X → (or X Y) | Sec : Y → (or X Y) .
```

The *selectors* Prim and Sec serve to identify the induction hypotheses of disjunction, two of them in this case. The vertical stroke in the definition of the parametrized or, serves to separate the hypotheses. Note that

```
or : Prop → Prop → Prop.
```

as required. In addition

```
Prim : (X, Y : Prop) X → (or X Y) .
Sec : (X, Y : Prop) Y → (or X Y) .
```

Facts

Suppose that we wish to apply this definition to verify the specific implication $(\text{or } A \ B) \rightarrow P$, where A , B , and P are declared as type Prop . Verification means that a member of the given expression P is to be constructed. The required verification is again achieved by *eliminating* the induction hypotheses, two of them in this case. As before the type of the ELIM expression must be P (this is significant). Because there are two λ abstractions involved the construction is a bit more complicated. Now there are two entities $p1$ and $p2$ to be constructed, both of them of type P .

```
[h : (or A B)] (ELIM h [\lambda a:A]p1 [\lambda b:B]p2).
```

Here the λ terms introduce a and b as local variables with their types. At the same time the terms refer to the two induction hypotheses (substituting A for X in the type of the selector Prim and B for Y in the type of the selector Sec). For the construction of $p1$ and $p2$, the available entities are A and B , the local a , and the local b , and in addition the selectors Prim and Sec (each of these two requiring three arguments).

An example (omitting writing λ 's) is

```
[h:(or A B)]
(ELIM h [a:A] (Sec B A a) [b:B] (Prim B A b) ) :
(or A B) → (or B A).
```

In the ELIM expression we keep as a rule fixed the sequence of induction hypotheses, as they appear in the inductive definition, i.e., the lexicographic sequence. In this example A and B of type Prop have been assumed to be declared, say as variables. But then it is clear that above example is valid for any A and B of that type. This commutative property of the disjunction connective can be expressed as

```
FACT f.2.3 : (x, y : Prop) (or x y) → (or y x).
```

The verification of this fact involving Prim and Sec is

```
f.2.3 =[x, y : Prop; h : (or x y)]
      (ELIM h [a:x] (Sec y x a) [b:y] (Prim y x b) ) :
      (or x y) → (or y x).
```

whence

```
(f.2.3 A B) : (or A B) → (or B A).
(f.2.3 B A) : (or B A) → (or A B).
```

We shall consider now an example of using the disjunction operator in an annotation of another text. It is a further elaboration of the example of a room containing books and shelves. Consider the sentences

```
a room R contains books and shelves S.
```

A is a book, the name is na, it is located on some shelf #1.
 B is a book, the name is nb, it is located on some shelf #2.
 C is a book, the name is nc, it is located on some shelf #2.

Here the books and the room are abstracted from the corresponding objects in the real world: the titles of the books are not written and the room is left unspecified. In order to be able to identify individuals (books in this case, room does not need to be regarded as an individual object in the same way as the others) and keeping them separated, they have been provided with names. Note that a book can be given two or more names, characterizing different aspects of the book. This is not contrary to L. Wittgenstein's *dictum*: "We do not give two names to one thing." On the other hand an entity which names a book cannot be the name of another book, in accordance with Wittgenstein's: "We do not give one name to two things." The shelves could also be given names, but that is not needed for the present exposé. Books B and C are located on the same shelf.

Let us see what annotation to these sentences, in the sense of LINCOS, could be given. To begin with declarations are needed.

```
CONSTANTS book, shelf : Set.
```

Books and their names

```
CONSTANTS A, B, C : book.  
CONSTANT na : A. CONSTANT nb : B. CONSTANT nc : C.
```

Shelves in the room

```
CONSTANTS #1, #2 : shelf.  
CONSTANT R : book → shelf → Prop.
```

Books have locations (on the shelves).

```
DEFINITION A-location: A → Prop := [x:A] (R x #1).  
DEFINITION B-location: A → Prop := [x:B] (R x #2).  
DEFINITION A-location: A → Prop := [x:A] (R x #2).
```

One expects the following to be the case.

```
FACT f.1.2.4 :  
((A-location na) ∨ (B-location nb)) → ((B-location nb) ∨ (A-location na)).
```

The verification is elegantly obtained by using fact f.2.3 as follows

```
f.2.4 = (f.2.3 (A-location na) (B-location nb)).
```

Connectives, Continued

Negation

We shall need the empty entity `nil` of type `Prop` (one might say the empty proposition). It is inductively defined by

```
INDUCTIVE nil : Prop := .
```

Note that this definition implies that `nil` is not given the possibility *a priori* of having residents—because of the absence of a selector following the `:=` token. The empty entity can be used to represent falsity. Using the empty entity we define logical negation by

```
DEFINE not : Prop → Prop := [A : Prop] A → nil.
```

`DEFINE` is often used instead of `DEFINITION` for easy reading by humans.

If `X` is some entity in `Prop`, then evidently $(\text{not } X) : \text{Prop}$ and $(\text{not } X) = X \rightarrow \text{nil}$.

We write usually $\sim X$ for $(\text{not } X)$. In the definition of `not` we have then

```
[A : Prop] A → nil : Prop → Prop
```

as expected. Examples of using negation are given in Part II.

Implication

The introduction rule for implication is

```
DEFINE imp : Prop → Prop → Prop := [A, B : Prop] A → B → Prop.
```

Assuming the types `X : Prop`, `Y : Prop`, note the following

```
imp : Prop → Prop → Prop
(imp X) : Prop → Prop
(imp X Y) : Prop.
```

We could have written `imp : (Prop → (Prop → Prop))` using the convention of brackets associating to the right in mappings.

For binary connectives we (might) also use infix notation, for easy reading by humans. Implication occupies a central position in LINCOS. The symbol $\rightarrow$, used in typing map definitions, is also used for logical implication, i.e., “`imp`” can be replaced by $\rightarrow$.

Note, however, that

```
DEFINE → : Prop → Prop → Prop := [A, B : Prop] A → B → Prop.
```

is *unacceptable*, even though it looks like a genuine recursive definition!

Suppose that $A \rightarrow B$ is the case and in addition that A is the case (in other words, it is given that for some a , A is the case because $a : A$). One expects then B to be the case. Can the fact

FACT f.2.5 : $(A \rightarrow B) \rightarrow B$.

be verified? In order to achieve this we need to construct an expression for f.2.5 of type $(A \rightarrow B) \rightarrow B$. Following the basic verification procedure in constructive logic we construct an abstraction of the correct type. That is obtained like this:

f.2.5 = $[H : A \rightarrow B] (H a)$.

Here H of type $A \rightarrow B$ is abstracted, while $(H a)$ is of course the application of H to a . In order to determine the type of f.2.5 represented by this expression, observe

$(H a) : B$.
f.2.5 = $[H : A \rightarrow B] (H a) : (A \rightarrow B) \rightarrow B$. QED.

We continue from here with brief discussions on the concepts of existence and equality in LINCOS, in fact the *existence* and *equality functions*.

Typing Existence and Equality

Not immediately needed here, but necessary to have at hand when discussing Aristotelian facts (Chap. 7) and elsewhere (Chap. 23), are existence functions in a restricted sense.

Existence

Consider the function Ex , parametrized by some X and some function $P : X \rightarrow \text{Prop}$, such that $(Ex X P)$ is the case. As the introduction rule specific for the case $X : \text{Set}$ we use

INDUCTIVE $Ex [X : \text{Set}; P : X \rightarrow \text{Prop}] : \text{Prop} :=$
Ex-intro: $(x : X) (P x) \rightarrow (Ex X P)$.

The induction hypothesis is seen to be $(x : X) (P x)$. The definition implies the following types of Ex and the selector Ex -intro (this is important)

$Ex : (X : \text{Set}) (X \rightarrow \text{Prop}) \rightarrow \text{Prop}$
 $Ex\text{-intro} : (X : \text{Set}; P : X \rightarrow \text{Prop}; x : X) (P x) \rightarrow (Ex X P)$.

Suppose that we wish to prove some implication $(Ex S Q) \rightarrow R$. That means that ELIM must be used to construct a resident r of R . There are two ways of representing the induction hypothesis under ELIM.

$[h : (Ex S Q)] (ELIM h [y : (x : S) (Q x)] r)$
 $[h : (Ex S Q)] (ELIM h [x : S; y : (Q x)] r)$

Sometimes which one is used makes a difference, sometimes not. Should we need an entity of type S “further along” it must be supplied externally in the first case while x can be used in the second case. When it makes no difference, we use the first case.

Three examples.

Suppose $s : S$, so $(Q\ s) : \text{Prop}$.

$$[h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [y : (x : S)\ (Q\ x)]h) : \\ (\text{Ex } S\ Q) \rightarrow (\text{Ex } S\ Q).$$

$$[h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [x : S; y : (Q\ x)]h) : \\ (\text{Ex } S\ Q) \rightarrow (\text{Ex } S\ Q). \text{ alternatively}$$

$$[h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [y : (x : S)\ (Q\ x)]y) : \\ (\text{Ex } S\ Q) \rightarrow (x : S)\ (Q\ x).$$

$$[h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [y : (x : S)\ (Q\ x)](y\ s)) : \\ (\text{Ex } S\ Q) \rightarrow (Q\ s).$$

The first example is a tautology. Moreover

FACT f.2.6 : $(\text{Ex } S\ Q) \rightarrow (\text{Ex } S\ Q)$.

is also verified by $\text{f.2.6} = [h : (\text{Ex } S\ Q)]h$.

The derived relation of the second example is remarkable. It is the result of accepting without restrictions $(\text{ELIM } h\ [y : (x:S)(Q\ x)]y)$. In Chapter 5 an alternative way of doing the elimination is explained and a justification for above elimination is given by means of a (curious) hypothesis. The derived relation above is sometimes referred to as “The Drinker’s Paradox” (*cf.* The Coq Project, referred to in Chap. 1). Suppose that S represents a bar, and $s : S$ a person in the bar. Further $(Q\ s)$ represents the fact that s is a drinker. The example states that if there is a drinker in the bar, all persons in the bar are drinkers (!). The pub is apparently a special one! Formally for $S : \text{Set}$ and $Q : S \rightarrow \text{Prop}$

FACT DP : $(\text{Ex } S\ Q) \rightarrow (x:S)\ (Q\ x)$

verified by

$$\text{DP} = [h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [y : (x:S)\ (Q\ x)]y).$$

The third result is also interesting. Note for instance that

FACT f.2.7 : $(\text{Ex } S\ Q) \rightarrow (Q\ s)$

is verified by

$$\text{f.2.7} = [h : (\text{Ex } S\ Q)]\ (\text{ELIM } h\ [y : (x:S)\ (Q\ x)](y\ s)).$$

The last result is only meaningful given a specific declaration of s . This aspect of *environmental information* is discussed in detail in Wittgenstein’s theatre (Chap. 8), in connection with the matter of verifications, the constructions involved, and their meaning.

Equality

The notion of equality as used in this treatise, designated by the $=$ symbol, is a meta notion, i.e., it is used to express “sameness” as seen from a level *outside of it*.

However, we are also interested in expressing “sameness” (also in the Leibniz’ sense) *within* the system. We use the function Eq in order to express this.

```
INDUCTIVE Eq [X : Prop; x : X] : X → Prop :=
  Eq-intro : (Eq X x x)
```

so that

```
Eq-intro : (X : Prop; x : X) (Eq X x x).
```

This is another case of an inductive definition without an induction hypothesis (the case of nil introduced in connection with negation is a special one). However, the selector, Eq-intro supplied with arguments can of course be used in applications. As an example consider the following fact expressing that equality over Prop is a transitive relation.

```
FACT f.2.8 : (X:Prop; x, y : X) ((Eq X x y) →
                                   (z : X) (Eq X y z) →
                                   (Eq X x z)) .
```

or alternatively

```
FACT f.2.8 : (X:Prop; x, y, z : X) ((Eq X x y) /\ (Eq
X y z)) → (Eq X x z) .
```

The verification is

```
f.2.8 = [X : Prop; x, y : X; h : (Eq X x y)]
        (ELIM h [z : X; h1 : (Eq X y z)] (ELIM h1 []
        (Eq-intro X x z)) ) .
```

In the first ELIM z and h1 are auxiliary local variables, in fact assumptions, to be used in the second ELIM. In the second ELIM there are no auxiliary variables and (Eq-intro X x z) has the correct type (Eq X x z).

That we have Leibniz equality is expressed by

```
FACT leibniz : (X : Prop; x,y : X; C : X → Prop)
               (Eq X x y) → (C x) → (C y) .
```

The verification is

```
leibniz = [X : Prop; x,y : X; C : X → Prop;
          h1 : (Eq X x y); h2 : (C x)]
          (ELIM h1 [h3: (C y)] h3 ) .
```

Booleans

Remembering nil : Prop, the Booleans are defined as follows

```
INDUCTIVE bool : Prop := false : bool | true : bool.
```

```

FACT is-false : bool → Prop.
is-false = [h : bool](ELIM h [] []nil []).

```

```

FACT is-true : bool → Prop.
is-true = [h : bool](ELIM h [] []nil []).

```

Remarks. Note that from $(\text{is-false false}) : \text{Prop}$ and $(\text{is-true false}) : \text{Prop}$ one *cannot* conclude that (is-true false) is the case or that (is-false true) is the case. So *false* and *true* do *not* represent falsity and truth. In fact *nil* is used to represent falsity, while there is no need to represent truth as we use the concept of verification in order to prove that something is the case. The defined *is-false* and *is-true* are introduced as type *markers*. In the definition of *bool* there are two empty induction hypotheses. In *ELIM* we can use for each of these local assumptions, but since we do not need those we use twice *[]*. For the case of the first hypothesis, *no* result type is assumed either, so we write *[]* followed by nothing. This is a special case of η reduction (Appendix D). For the case of the second hypothesis, we write *[]nil*, so that the result type is *Prop*.

By applications of Modus Tollens (i.e., $\text{MT} : (\sim B \wedge (A \rightarrow B)) \rightarrow \sim A$ discussed in some detail in Chap. 5), we find the facts

```

FACT f.2.9 :
  (~ (Eq bool false true) /\ ((is-false true) →
    (Eq bool false true)) ) →
  ~(is-false true).

```

```

FACT f.2.10 :
  (~ (Eq bool true false) /\ ((is-true false) →
    (Eq bool true false)) ) →
  ~(is-true false).

```

And of course also

```

FACT f.2.11 :
  ~(is-false true) /\ ((Eq bool false true) →
    (is-false true)) →
  ~(Eq bool false true).

```

```

FACT f.2.12 :
  ~(is-true false) /\ ((Eq bool true false) →
    (is-true false)) →
  ~(Eq bool true false).

```

Next we show that the selectors *false* and *true* are unequal.

```

FACT uneq-false-true : ~(Eq false true).
uneq-false-true = (leibniz bool is-false false true).

```

This results from

```

(leibniz bool is-false false true) :
  (Eq false true) → nil.

```

or

```
(leibniz bool is-false false true) : ~(Eq false true).
```

In the same way we find that the selectors true and false are unequal.

```
FACT uneq-true-false : ~(Eq true false).
```

```
uneq-true-false = (leibniz bool is-true true false).
```

Note that a binary coding lies behind the formalism employed here. The sequence of selectors in the inductive definition of bool is false followed by true, and is-false is associated with `[]nil []`, and is-true is associated with `[] []nil` via ELIM. This observation supplies a key to the case of more than two selectors in an inductive definition. This can be illustrated by

```
INDUCTIVE value : Prop :=
  maybe-false : value | maybe-true : value |
  maybe-not : value.
```

Now we need three markers

```
FACT is-maybe-false : value → Prop.
```

```
is-maybe-false = [h : value](ELIM h []nil [] []).
```

```
FACT is-maybe-true : value → Prop.
```

```
is-maybe-true = [h : value](ELIM h [] []nil []).
```

```
FACT is-maybe-not : value → Prop.
```

```
is-maybe-not = [h : value](ELIM h [] [] [] []nil).
```

By means of the mechanism explained one can verify that any two selectors occurring in this inductive definition are unequal: leibniz is applied. For example, showing that maybe-false and maybe-not are unequal, is done like this

```
FACT uneq-maybe-false-maybe-not :
  ~(Eq maybe-false maybe-not).
uneq-maybe-false-maybe-not =
  (leibniz value is-maybe-false maybe-false
   maybe-not).
```

This is because

```
(leibniz value is-maybe-false maybe-false maybe-not) :
(Eq maybe-false maybe-not) → nil.
```

or

```
(leibniz value is-maybe-false maybe-false maybe-not) :
~(Eq maybe-false maybe-not).
```

Chapter 3

Hypotheses

Intention

In defining entities in a *Lingua Cosmica* based on the calculus of constructions CC and the calculus of constructions with induction CCI great care must be taken in building the so-called *environment*. This object is a collection of entities either built from introductory declarations, or derived conclusions, but it can contain also *hypotheses*. An environment should be consistent, i.e., free of contradictions. Experiments and experience in building environments have shown that inconsistencies can easily occur—but are not always easy to detect and may have unwanted consequences (see especially Chap. 24). The present chapter illustrates some simple pitfalls a designer can tumble in as a result of introducing hypotheses. It also shows, however, that one cannot always avoid the use of hypotheses. This is illustrated by a rather amusing case: a section from Lewis Carroll *Alice's Adventures in Wonderland* in Chap. 9 *The Mock Turtle Story* (Lewis Carroll 1865) containing double negations.

Contradictions

Sometimes rules, written as implications, can be introduced in the environment as hypotheses, but this should be done with great care in order to avoid contradictions. The simplest case of a contradiction is that of a set of declarations of constants from which the empty entity can be derived. Let

```
CONSTANTS x, y : Prop.  
CONSTANT h0 : x.  
CONSTANT h1 : x → y.  
CONSTANT h2 : ~x.
```

Where $\sim : \text{Prop} \rightarrow \text{Prop}$. This environment is not sound, as seen informally by considering h_2 contradicting h_0 , or the other way around. The contradiction is formally derived as follows. First note the equalities

$\sim x = (\text{not } x) = x \rightarrow \text{nil}.$

This is because in Chap. 2 we have given the definition of negation

DEFINE not : Prop $\rightarrow$ Prop := [a:Prop]a $\rightarrow$ nil

with for the entity nil as before

INDUCTIVE nil : Prop :=.

Take notice

not : Prop $\rightarrow$ Prop.

$\sim : \text{Prop} \rightarrow \text{Prop}.$

(not x) = x $\rightarrow$ nil : Prop.

$\sim x = x \rightarrow \text{nil}.$

(not (not x)) = (x $\rightarrow$ nil) $\rightarrow$ nil : Prop.

$\sim \sim x = (x \rightarrow \text{nil}) \rightarrow \text{nil}.$

Then note that x is the case in the environment because $h_0 : x$. However, $h_2 : \sim x$, i.e., (not x) is also the case—do we have here a contradiction? In order to find how a contradiction like this one is concluded, consider two possible applications. In the environment $h_1 : x \rightarrow y$ can be applied to $h_0 : x$, giving $(h_1 h_0) : y$. However, $h_2 : x \rightarrow \text{nil}$ can be applied to h_0 as well giving

$(h_2 h_0) : \text{nil}$. Here nil represents *falsity* pinpointing the contradiction in the environment.

Note that nil has no residents—but can be given one, for instance by defining

CONSTANT void : nil.

If x is the case (i.e., $h_0 : x$ as defined above), then FACT $h_3 : x \rightarrow \text{nil}$, can be verified. This is because $h_3 = [h_0 : x]\text{void} : x \rightarrow \text{nil}$. So we have here the similar contradiction as before, *viz.*

$(h_3 h_0) : \text{nil}$. So far so good. If, however, the environment is enriched carelessly by the hypothesis

HYPOTHESIS oddity : nil $\rightarrow$ (X:Prop) X.

then if nil is the case because void : nil, then (oddity void) : (X : Prop) X, so (X : Prop)X is the case, i.e., every X is the case. Since as said, nil can be regarded as representing falsity, we have here *ex falso quodlibet*. A useful hypothesis in this respect is the absurdity hypothesis

HYPOTHESIS absurd : (X, Y : Prop) (X $\rightarrow$ $\sim$ X $\rightarrow$ Y).

Double negation

In Chap. 6 De Morgan's laws and the two rules for double negation are introduced as hypotheses. Consider the double negation rules written as facts for some $A : \text{Prop}$

FACT DN1 : $\sim \sim A \rightarrow A$.
 FACT DN2 : $A \rightarrow \sim \sim A$.

Referring to Chap. 8 (Wittgenstein's Theatre) we state here without specifying the argument that DN1 cannot be verified in LINCOS. We might therefore introduce this rule in a general form as a hypothesis

HYPOTHESIS DN1 : $(x : \text{Prop}) (\sim \sim x \rightarrow x)$.

In the case of the second double negation rule DN2 we can write the verification of the fact as follows

DN2 = $[h1 : A] ([h2 : A \rightarrow \text{nil}] (h2 \ h1))$

because $[h1 : A] ([h2 : A \rightarrow \text{nil}] (h2 \ h1)) : A \rightarrow (A \rightarrow \text{nil}) \rightarrow \text{nil}$

where $A \rightarrow (A \rightarrow \text{nil}) \rightarrow \text{nil} = A \rightarrow ((A \rightarrow \text{nil}) \rightarrow \text{nil})$ due to the property that brackets associate to the right. Finally

$A \rightarrow ((A \rightarrow \text{nil}) \rightarrow \text{nil}) = A \rightarrow \sim \sim A$.

A little more complicated case containing a double negation is

FACT DN : $(x : \text{Prop}) (\sim (x \ / \ \sim x))$.

In order to verify DN we need to construct an entity of type $(x : \text{Prop})((x \wedge (x \rightarrow \text{nil})) \rightarrow \text{nil})$.

Here is the construction.

DN = $(x : \text{Prop}) [h : (x \ / \ (x \rightarrow \text{nil}))]$
 $(\text{ELIM } h \ [h1 : x; h2 : x \rightarrow \text{nil}] (h2 \ h1))$.

Next we consider a rather amusing case: a section from *Alice's Adventures in Wonderland* (Lewis Carroll 1865) in Chap. 9 *The Mock Turtle Story* containing double negations.

"I quite agree with you," said the Duchess; "and the moral of that is—'Be what you would seem to be'—or if you'd like it put more simply—'Never imagine yourself not to be otherwise than what it might appear to others that what you were or might have been was not otherwise than what you had been would have appeared to them to be otherwise.'"

"I think I should understand that better," Alice said very politely, "if I had it written down: but I'm afraid I can't quite follow it as you say it."

"That's nothing to what I could say if I chose," the Duchess replied, in a pleased tone.

"Pray don't trouble yourself to say it any longer than that," said Alice.

“Oh, don’t talk about trouble!” said the Duchess. “I make you a present of everything I’ve said as yet.”

“A cheap sort of present!” thought Alice. “I’m glad they don’t give birthday presents like that!” But she did not venture to say it out loud.

Alice upon hearing the Duchess’s spoken advice would like to see the message written down, so that it might be given an interpretation. Above is in fact a written version—using LINCOS we shall give it the interpretation Alice wished to see. First we show a little more structure in the story.

The Duchess’s advice is

1. “Be what you would seem to be” or
2. “*Never* imagine yourself *not* to be otherwise than what it might appear to others (that what you were *or* might have been was *not* otherwise than what you had been) would have appeared to them to be otherwise.”

Consider first (1).

```

CONSTANT Alice : Prop.
CONSTANT be : Prop → Prop → Prop.
CONSTANT what-you-seem-to-be : Prop → Prop.
Note (be Alice (what-you-would-seem-to-be Alice)) : Prop.
"Be Alice what you would seem to be Alice"

```

Then go to (2). In order to simplify the advice we replace the first line of (2) by the (perhaps not quite grammatical) line.

“Imagine yourself Alice to be *never not* otherwise than”

```

CONSTANT imagine-yourself-to-be : Prop → Prop → Prop.
CONSTANT otherwise-than : Prop → Prop.

```

Note

```

(x : Prop) (imagine-yourself-to-be Alice (otherwise-
than x)) : Prop.
FACT f.3.1. : (x : Prop) (~ ~ (otherwise-than x) →
(otherwise-than x)).
f.3.1 = (x : Prop) (DN1 (otherwise-than x))
because for any x : Prop
(DN1 (otherwise-than x)) : (~ ~ (otherwise-than x) →
(otherwise-than x)).

```

So we may use

"Imagine yourself Alice to be otherwise than"

Next we wish to simplify
 (that what you were *or* might have been was *not* otherwise than what you had been)

by omitting “was not otherwise than.”

```

CONSTANT what-you-were : Prop → Prop.
CONSTANT what-you-might-have-been : Prop → Prop.
CONSTANT was-not-otherwise-than : Prop.
CONSTANT what-you-had-been : Prop → Prop.

```

This can be effectuated by using equality over Set

```

INDUCTIVE Eq : [X : Set; x : X] : X → Prop :=
Eq-intro : (Eq X x x)

```

and formulating the hypothesis

```

HYPOTHESIS replace :
(Eq Prop
((what-you-were Alice) \ / (what-you-might-have-been
Alice)) →
was-not-otherwise-than → (what-you-had-been
Alice) (* arg 1 *))
((what-you-were Alice) \ / (what-you-might-have-been
Alice)) →
(what-you-had-been Alice) (* arg 2 *))

```

In view of the hypothesis we may use

that what you were or might have been, was what you had been

Left over is now

```

CONSTANT what-it-might-appear-to-others : Prop.

```

what it might appear to others

```

CONSTANT would-have-appeared-to-them-to-be-otherwise : Prop

```

would have appeared to them to be otherwise.

Combining the foregoing we find after slightly paraphrasing the text and using

```

CONSTANT be : Prop → Prop → Prop.
or HYPOTHESIS be : Prop → Prop → Prop.
CONSTANTS a, b : Prop.

```

“Imagine yourself Alice to be otherwise ((be Alice) ~a)

what it might appear to others

that ..., was what you had been ((be Alice) a)

would have appeared to them to be otherwise” ((be Alice) ~a).

That concludes the discussion on the use of double negations.

Hypotheses

Hypotheses can be useful but the introduction of them must be done with care. Consider the two terms $(A \rightarrow (A \rightarrow B)) \rightarrow B$ and $((A \rightarrow B) \rightarrow A) \rightarrow B$ for $A, B : \text{Prop}$. In classical logic

$(A \rightarrow (A \rightarrow B))$ is equivalent to $A \rightarrow B$ and $((A \rightarrow B) \rightarrow A)$ is equivalent to A . So we find the classical equivalences

$(A \rightarrow (A \rightarrow B)) \rightarrow B$ equivalent to $(A \rightarrow B) \rightarrow B$
 $((A \rightarrow B) \rightarrow A) \rightarrow B$ equivalent to $A \rightarrow B$.

The two terms are not verifiable as facts in LINCOS. Consulting Wittgenstein's theatre (Chap. 8), it is easily seen that the verification machinery for facts gets stuck in both cases (because A cannot be justified). What if they are placed in the environment as hypotheses? (See also Chap. 24 for a discussion of this particular case in a different and broader context).

HYPOTHESIS H1 : $(A \rightarrow (B \rightarrow A)) \rightarrow B$.
 HYPOTHESIS H2 : $((A \rightarrow B) \rightarrow A) \rightarrow B$.

Consider first hypothesis H1. We have

$[h1 : A, h2 : B] h1 : A \rightarrow B \rightarrow A$

so that $A \rightarrow B \rightarrow A$ or $A \rightarrow (B \rightarrow A)$ is the case. This implies the relation

$(H1 [h1 : A, h2 : B] h1) : B$, i.e., B is the case.

Consider next hypothesis H2. We have

$[* : [h1 : A, h2 : B]] h1 : (A \rightarrow B) \rightarrow A$

so that $(A \rightarrow B) \rightarrow A$ is the case. But then $(H2 [* : [h1 : A, h2 : B]] h1) : B$, i.e., B is the case.

Since we have not assumed a priori that B is the case, we have reached here in both cases contradictions. Moreover the result that B is the case is most awkward: it means that even though B is not explicitly supplied with an inhabitant, B has one!!

Reference

Lewis Carroll *Alice's Adventures in Wonderland* (1865)

Chapter 4

Higher Orders and Inductive Structures

Intention

The design of a cosmic language for interstellar communication should satisfy various significant requirements. Power of expression, the theme of the present chapter, is one of them. This chapter is in a way a continuation of Appendix D, because we present here to begin with interpretations of the combinatory functions I, K and S discussed in that chapter. The interpretations are in terms of LINCOS conventions explained already, notably the concept of inductive definitions. Laws over the combinatory functions are easily constructed using these conventions. This part of the chapter is followed by an in-depth discussion on the concept of recursion, using an important inductive structure, a Matrjoshka—матрёшка -, the well-known Russian doll. Treating this non-elementary case is justified because it illustrates the power of expression using inductive structures. At the same time there are the following considerations.

The *Lingua Cosmica* treated in this treatise is a (meta) system primarily aimed at the task of annotating contents of possibly large-scale messages for ETI. As mentioned LINCOS is based on formal constructive logic. It is designed for dealing with logic contents of messages but it is also applicable for denoting structural properties of more general abstractions embedded in such messages. The second part of the present chapter explains ways and means for achieving this for a special case: inductive or recursive entities.

In order to appropriately treat and illustrate recursion, i.e. the way the concept is applied and the difficulties that can be encountered, two stages are involved: as usual first the domain of discourse is enriched with suitable representations of the entities concerned, after which properties over them can be dealt with *within* the system itself.

Readers are advised to skip most of this chapter at first reading and turn to the more elementary descriptions and discussions in the beginning of PART II.

The Combinators

Functions taking functions as arguments are *high order functions*. Examples of these are the combinators I, K and S discussed in Appendix D. We start with naïve definitions, now formulated with LINCOS conventions.

```

DEFINE I : Set → Set := [x:Set]x.
DEFINE K : Set → Set → Set := [c : Set, * : Set]c.
DEFINE S : (Set → Set → Set) → (Set → Set) → Set
→ Set :=
    [f : Set → Set → Set; g : Set → Set;
     x : Set](f x (g x)).

```

Here the entities c , f , g and x are lambda bound, and implicit (one could also say that they are formal variables). We can, however, also explicitly declare four constants c , f , g and x .

```

CONSTANT f : Set → Set → Set.
CONSTANT g : Set → Set.
CONSTANT c : Set.
CONSTANT x : Set

```

and then as expected

```

(S f g x) : Set and (S f g x) = (f x (g x)).
(S K I x) : Set and (S K I x) = (K x (I x)).

```

As $(I x) = x$ we expect also $(S K I x) = (K x x) = x$, how is that to be shown? Besides we need to show something like

```

(S K I (K x x)) = x

```

as well. So instead we change to inductive definitions for I, K and S as follows.

```

INDUCTIVE I [x : Set] : Set := i-intro : (I x).
INDUCTIVE K [x, y : Set] : Set := k-intro : x.
INDUCTIVE S [f : Set → Set → Set; g : Set → Set;
x : Set] : Set :=
    s-intro : (S f g x).

```

These definitions imply for the types of the selectors

```

i-intro : (x : Set)(I x).
k-intro : (x, y : Set)x.
s-Intro : (f : Set → Set → Set; g : Set → Set;
x : Set)(S f g x).

```

Note the lambda bindings of f , g and x . If these variables would be globally bound by

```

CONSTANT f : Set → Set → Set.
CONSTANT g : Set → Set.
CONSTANT x : set

then

(s-intro f) : (g : Set → Set; x : Set) (S f g x).
(s-intro f g) : ( x : Set) (S f g x).
(s-intro f g x) : (S f g x).

and

(S f g x) : Set.
(S f (f x) (f x x)) : Set.

```

Note that in the definitions of I, K and S the induction hypotheses are empty, but there are variables associated with the selectors. In order to verify relations like $(S f g x) \rightarrow (f x (g x))$ and the other way around, we must construct facts. For conceptual aspects of facts and proof theory, see the detailed discussions in Part II. Here we use the elimination technique explained in Chap. 2. ELIM is used to get hold of the variables f, g and x associated with the selector s-intro.

```

FACT f.4.1 : (S f g x) → (f x (g x)).
[H : (S f g x)] (ELIM H
  [f : Set → Set → Set;
   g : Set → Set;
   x : Set]) (f x (g x)) =
  (S f g x) → (f x (g x)).

```

In order to prove $(f x (g x)) \rightarrow (S f g x)$ we need inductive definitions for f and g in the same way as given for S.

```

INDUCTIVE f [x, y : Set] : Set := f-intro : (f x y).
INDUCTIVE g [x : Set] : Set := g-intro : (g x).

```

For the selectors

```

f_intro : (x,y :Set) (f x y).
g_intro : (x : Set) (g x)

```

and as we have seen s-Intro : $(f : Set \rightarrow Set \rightarrow Set; g : Set \rightarrow Set; x : Set)(S f g x)$.

```

FACT f.4.2 : (f x (g x)) → (S f g x).
[H : (f x (g x))] (ELIM H
  [h1 : f;
   h2 : x;
   h3 : (g x)] (s-intro h1 (h3 h2)
  h2)) = (f x (g x)) → (S f g x).

```

Other relations as $(S K I x) = (K x (I x))$ can be proved in analogous ways. We will leave the discussion on high order functions now and consider the concept of recursion next.

Recursion is mentioned briefly (for the case of the faculty function) already in Appendix D, but is extended here in a more general, structural sense. Readers are strongly advised to skip the following section at first reading. We consider here recursion in a general sense as it occurs in the well-known Russian dolls, see also the paper (Ollongren 2011). The present section is a revised version of that paper. A more elementary description of a Russian doll is in Chap. 11.

Bounded Matrjoshka

В кукле кукла

Кукла в кукле

As a characteristic example of an inductive structure the case of Russian dolls (куклы), dolls within dolls, is chosen. These objects are rather simple in because they have a linear one-dimensional recursive structure. Rather loosely stated one might say that a matrjoshka contains in itself a matrjoshka (*матрёшка заключает в себе матрёшку*)—or that a matrjoshka is contained in a matrjoshka. In the present chapter we consider first a restricted sequence of dolls, i.e. the case of the bounded matrjoshka. Over the sequence the concepts “contains” as well as “is contained in” can be modelled in Lingua Cosmica. In both cases the grammatical *imperfective aspect* is used in Russian. In the case of the dolls the concept expressed by imperfective verbs *заключать* or *содержать* corresponding with “to contain”, is *transitive*. It propagates unchanged from the outside inwards and from the inside outwards in matrjoshka’s. Using the conventions of LINCOS we begin by specifying the domain of discourse (the environment) by defining the types of the objects needed, as follows.

```

CONSTANTS d1, d2, d3 : Set. (* individual dolls *)

DEFINE contain : Set → Set → Prop := [x, y : Set] (x → y).
DEFINE contain-trans : Set → Set → Set → Prop :=
    [x, y, z : Set] (contain x y) /\
    (contain y z) → (contain x z).

DEFINE contained-in : Set → Set → Prop :=
    [y, x : Set] (y → x).
DEFINE contained-in-trans : Set → Set → Set → Prop :=
    [x, y, z : Set]
    (contained-in z y) /\
    (contained-in y x) →
    (contained-in z x).
```

Consider now a concrete matrjoshka consisting of the three individual dolls, d1, d2 and d3. In the environment we introduce new types m1, m2, n1 and n2 registering configurations as follows

```
CONSTANTS m1 : (contain d1 d2), m2 : (contain d2 d3).
CONSTANTS n1 : (contained-in d3 d2),
          n2 : (contained-in d2 d1).
```

Given this environment facts as these can be concluded

```
FACT f.4.3 : (contain d1 d3).
FACT f.4.4 : (contained-in d3 d1).
```

Verifications of these facts are achieved by constructing expressions for f.4.3 and f.4.4 of the correct type. We find

```
f.4.3 = (contain-trans d1 d2 d3) : (contain d1 d3).
f.4.4 = (contained-in-trans d3 d2 d1) : (contained-in
d3 d1).
```

Unbounded matrjoshka. Above discussion can be extended to larger matrjoshka's, but that is not useful for the general case we consider now, i.e. when it is not known in advance how many dolls "are in the picture". So we set up a new environment to deal with the more general case. At the same time we show how to treat recursion in a uniform manner. We start (anew) by enriching the environment with an inductive definition:

```
INDUCTIVE Matr : Prop := Doll : Matr | S : Matr → Matr.
```

using two *selectors* Doll and S. Note, in view of the types in the definitions, we find the following justifications

```
(S Doll) : Matr, (S (S Doll)) : Matr, ....
```

We have here an unbounded sequence of dolls. Over the sequence the relation "contain" as well as "is contained in" is admissible. Informally: Doll contains (S Doll), which contains

(S (S Doll), ... and (S (S Doll)) is contained in (S Doll), which is contained in Doll, ...

S is interpreted as the successor function. Both relations are transitive. For example, writing (S² Doll) for (S(S Doll)) etc., we have

```
if, for n>0, (Sn Doll) : Matr contains (Sn+1 Doll) :
Matr which contains (Sn+2 Doll) : Matr,
```

```
then (Sn Doll) : Matr contains (Sn+2 Doll) : Matr.
```

Associated with above definition of Matr is the so-called *inductive type*

```
Matr-ind : (P : Matr → Prop) (P Doll) →
  ((x : Matr) (P x) → (P (S x)) → (x : Matr)
  (P x)).
```

Note that we use the suffix *-ind* to emphasize that the type is a new one associated with *Matr*.

This type is a parametrization of *Matr* (giving it a new name) with some property *P*, to be specified in applications. It consists of a sequence of two induction hypotheses

```
(P : Matr → Prop) (P Doll) and
(x : Matr) (P x) → (P (S x))
```

followed by the conclusion $(x:\text{Matr})(P\ x)$. According to constructive logic (the basis of LINCOS) the conclusion is the case for all *matrjoshkas*, if a *resident* of $(x : \text{Matr})(P\ x)$ can be constructed. In other words—if $(x : \text{Matr})(P\ x)$ *is the case*. In order to achieve this, the first and second induction hypotheses must be the case; this amounts to finding residents of the first and second induction hypothesis—so that they can be *eliminated*.

We show by an example how a conclusion can be achieved. Informally it will be shown that if the “first” doll is coloured, then all dolls are coloured. For *P* we substitute *is-coloured*. We need a constant and a definition

```
CONSTANT colour : Prop.
DEFINE is-coloured : Matr → Prop:=
[x : Matr] colour.
```

For the latter we can also write

```
DEFINE is-coloured (x : Matr) : Prop:= colour.
```

The definition implies $(\text{is-coloured Doll}) : \text{Prop}$, but it does *not* mean that the first doll is coloured. So we need to state explicitly that the “first” object, i.e. *Doll*, is coloured, by a hypothesis (to be used for eliminating the first induction hypothesis):

```
HYPOTHESIS Start : (is-coloured Doll).
```

This assumption says that $(\text{is-coloured Doll})$ is the case. Next we state that if a doll is coloured, then the successor doll is also coloured (to be used for eliminating the second induction hypothesis):

```
HYPOTHESIS Succ : (x : Matr) (is-coloured x) →
(is-coloured (S x)).
```

Note the subtlety that this hypothesis does *not* (yet) imply that $(x : \text{Matr}) (\text{is-coloured } x)$ is the case. It just says that if for some *a*, $a : (x : \text{Matr}) (\text{is-coloured } x)$, then $(\text{is-coloured } (S\ x))$ is the case, because $(\text{Succ } a) : (\text{is-coloured } (S\ a))$. Using the two hypotheses we can conclude that *all* dolls in the unbounded *matrjoshka* are coloured.

```
FACT f.4.5 : (x : Matr) (is_coloured x)).
```

The verification is rather easy. Observe first, substituting *is-coloured* for *P* in the definition of *Matr-Ind*,

that we have the relation

$$\begin{aligned} (\text{Matr-ind is-coloured}) : & (\text{is-coloured Doll}) \rightarrow \\ & ((x : \text{Matr})(\text{is-coloured } x) \rightarrow \\ & (\text{is-coloured } (S \ x))) \rightarrow \\ & (x : \text{Matr})(\text{is-coloured } x). \end{aligned}$$

Further

$$\begin{aligned} (\text{Matr-ind is-coloured Start}) : & ((x : \text{Matr})(\text{is-coloured } x) \rightarrow \\ & (\text{is-coloured } (S \ x))) \rightarrow \\ & (x : \text{Matr})(\text{is-coloured } x) \end{aligned}$$

and then

$$\begin{aligned} (\text{Matr-ind is-coloured Start Succ}) : & (x : \text{Matr}) \\ & (\text{is-coloured } x) \end{aligned}$$

leading to

$$f.4.5 = (\text{Matr-ind is-coloured Start Succ}).$$

Syntactic Structures

The more complicated case of recursive structures embedded within recursive structures (linear multidimensional recursion) is common in natural language expressions, i.e. written or spoken sentences. A non-empty sentence (Sent) can contain either just a number of terminals or a noun part (NounP) followed by a (possibly empty) verbal part (VerbP) surrounded by sequences of terminals. Terminals are words from a vocabulary. Let α , β and γ be sequences of terminals (possibly of length zero), ε is the empty sequence (of length zero). We write for the above

$$\text{Sent} \Rightarrow \alpha \text{ NounP } \beta \text{ VerbP } \gamma.$$

Sentences are recursive because the noun part can contain a sentence. The same applies to the verbal part. We write

$$\text{NounP} \Rightarrow \text{Sent} \mid \varepsilon, \text{VP} \Rightarrow \text{Sent} \mid \varepsilon.$$

These rules combine to

$$\text{Sent} \Rightarrow (\alpha \mid \text{Sent1}) (\beta \mid \text{Sent2}) \gamma,$$

with $\text{Sent1} \Rightarrow \alpha \rightarrow \text{Sent}$ and $\text{Sent2} \Rightarrow \beta \rightarrow \text{Sent}$.

In LINCOS this is represented by

INDUCTIVE Sent : Prop :=

$$\begin{aligned} s1 : & \alpha \beta \gamma \rightarrow \text{Sent} \mid \\ s2 : & \alpha \rightarrow \text{Sent2} \rightarrow \gamma \rightarrow \text{Sent} \mid \\ s3 : & \text{Sent1} \rightarrow \beta \gamma \rightarrow \text{Sent} \mid \\ s4 : & \text{Sent1} \rightarrow \text{Sent2} \rightarrow \gamma \rightarrow \text{Sent} \end{aligned}$$

```

with
INDUCTIVE Sent1 : Prop :=
  s5:  $\alpha \rightarrow \text{Sent} \rightarrow \text{Sent1}$ 
  WITH INDUCTIVE Sent2 : Prop :=
    s6 :  $\beta \rightarrow \text{Sent} \rightarrow \text{Sent2}$ .

```

We have here the recursive structure *Sent* with *Sent1* and *Sent2*, also recursive, *embedded* in it. In other words the inductive definition of *Sent* contains subsidiary definitions which are also inductive.

It is seen that a one-dimensional *matrjoshka* over the terminal *Doll* is a concretization of a special syntactic structure (α , β and γ empty, VerbP absent) with only one embedding because *Sent2* is absent (and so *s2*, *s4* and *s6* are absent). Substituting *Matr* for *Sent* and *matr1* for *Sent1*, and furthermore *Doll* for *s1*, *S* for *s3* and keeping *s5*, we have immediately the less simple case

```

INDUCTIVE Matr : Prop :=
  Doll : Matr | S : matr1  $\rightarrow$  Matr
  WITH INDUCTIVE matr1 : Prop :=
    s5 : Matr  $\rightarrow$  matr1.

```

Because of the interaction between *Matr* in the main part of the definition and the subsidiary *matr1* (i.e. the “with” part of the definition), we get now a more involved inductive form associated with the definition.

```

Matr-ind : (P : Matr  $\rightarrow$  Prop) (P Doll)  $\rightarrow$ 
  (x : Matr) (.find y : matr1 s.t. (S y) :
    Matr in (P (S y)) .)  $\rightarrow$  (x : Matr) (P x).

```

In order to satisfy the second inductive part defining *matr1*, *y* must be constructed *s.t.* (i.e. such that) (S *y*) : *Matr*. For finding *y* in this case only the definition of *Matr* is available. We observe that if *x* : *Matr* then (S (*s5* *x*)) : *Matr* and conclude *y* = (*s5* *x*). So above type of *Matr-ind* is resolved into

```

Matr-ind : (P : Matr  $\rightarrow$  Prop) (P Doll)  $\rightarrow$ 
  (x : Matr) (P (S (s5 x)))  $\rightarrow$  (x : Matr) (P x).

```

Next consider inductive forms associated with the more general syntactic structure of sentences in natural languages. As generic example we use the more extensive inductive definition of sentences introduced above, in the form of two-dimensional *matrjoshkas*.

```

INDUCTIVE Matr : Prop :=
  Doll : Matr | s2 : matr2  $\rightarrow$  Matr | S : matr1  $\rightarrow$ 
  Matr | s4 : matr1  $\rightarrow$  matr2  $\rightarrow$  Matr
  WITH INDUCTIVE matr1 : Prop :=
    s5 : Matr  $\rightarrow$  matr1 WITH INDUCTIVE matr2 :
    Prop := s6 : Matr  $\rightarrow$  matr2.

```

The inductive form associated with this definition is

```

Matr-ind : (P : Matr → Prop) (P Doll) →
  (x : Matr) ( . (.. find z : matr2 s.t.
    (s2 z) : Matr in (P (s2 z)) ..) →
    (.. find y : matr1 s.t. (S y) : Matr in (P
      (S y)) ..) →
    (.. find y : matr1, z : matr2 s.t.
      (s4 y z) : Matr in (P (s4 y z))..) .) →
  (x : Matr) (P x) .

```

In order to resolve this form we try to identify z with $(s6\ Doll) : matr2$ and y with $(s5\ Doll) : matr1$. This indeed does the resolution because $(s2\ z) : Matr$ and $(S\ y) : Matr$.

As a result we are left with the reduced form

```

Matr-ind : (P : Matr → Prop) (P Doll) →
  (x : Matr) ( (P (s2 (s6 Doll))) →
    (P (S (s5 Doll))) →
    (P (s4 (s5 Doll)) (s6 Doll)) ) →
  (x : Matr) (P x) .

```

Above two examples show that, given an inductive definition, the associated inductive form is easily found. However, in order to resolve an inductive form (i.e. finding appropriate expressions for the local variables so that they are eliminated) *computation* is required, essentially by matching procedures. Such computation may include iteration and is often non-trivial. In the present chapter we don't elaborate on the ways and means available for finding the required matches.

The preceding discussion illustrates in which way recursive structures in LINCOS are treated. In the environment (mainly type definitions of constants, variables and maps) recursive entities may occur in the form of inductive definitions of maps in terms of themselves. Associated with a recursive map is an inductive form, possibly with local unresolved variables (i.e. variables not already in the global environment). Assuming the occurring recursive entities to be well-founded*, the inductive forms can be resolved by matching procedures into *bona fide* types. Iteration may be needed. The resolved types can then be injected into the environment—enriching it.

*A recursive entity is *well-founded* if it admits a partial ordering with a bottom element. In the various *matrjoshkas* discussed above this is the case with the (resolving) bottom element usually *Doll*, but also $(s5\ Doll)$, $(s6\ Doll)$ and $(s5\ x)$ for any x of type *Matr* qualify and can be used. In this case the *find* variables can be bound to expressions equal to them.

The notion of well-foundedness may be fundamental also in cosmology, more in particular in the history of the Universe. The 0-point explosive event (the Big Bang, more or less a misnomer) is generally accepted as the beginning of our Universe. It is followed by expansion, and there are several possible scenarios for the development. One of them, attributed to David Penrose, is that there is *no* end to the history

of the Universe which will exist into infinity, with diminishing density of fundamental particles (never reaching 0). Supposing the Universe never to diminish in size, the conclusion is that the Universe is well-founded.

Reference

A. Ollongren *Recursivity in Lingua Cosmica*, Acta Astronautica 68 (2011) 544–548

Part II

Facts

The facts in logical space are the world.

L. Wittgenstein, *Tractatus Logico Philosophicus*, 1.13, 1921.

Introduction

A necessary (but not sufficient) condition for recipients of interstellar messages to be able to understand a message written in the LINCOS representation system and transmitted as a linear sequence of tokens (symbols), is that they are familiar with the basics of the calculus of constructions. In that case they (ETI) will hopefully be able to transliterate incoming sequences to their own representations. In addition to this the receiving ETI will need to supply some kind of interpretation to symbols and combinations of them used in our astrolinguistic messages. Hopefully the significance of designators as CONSTANT, HYPOTHESIS, FACT, etc., will be recognized. They are part of what can be called the *signature* of LINCOS (Chap. 16). Clearly a multitude of examples must be provided in messages, i.e., we are concerned with large-size messages. Contrary to mathematical practice better much redundancy in this respect than none or nearly none.

If some text written in a natural human language is used at the first level (often a preferred case considered in this book), the annotations part of the message is, as explained, concerned with the logical contents of the text—or possibly only of parts of the text. So the annotations should represent the logical meaning of the text. For that purpose we use in LINCOS concepts and constructions explained in Part I of the present treatise, i.e., declarations, definitions, hypotheses, and facts, collected in a logic database, also called an environment. The occurring facts are deductible by constructive means from available information in the base, using essentially simple tools. We have seen that one of the most important means available is lambda abstraction.

The question presents itself now what can be said about the meaning of facts themselves. We owe the view adopted here (and throughout the treatise) to L. Wittgenstein's: "*Der Sinn des Satzes ist seine Verification*". Transposed to the LINCOS system this would mean that facts should be accompanied by their verifications, the formal proofs.

The examples provided hitherto have already shown that verifications are usually lengthy. Is that an argument for giving proofs a lesser *status* in the annotations part? The sizes of proofs or the intricacies often present would be one reason for giving them less attention or even considering omitting them in messages for ETI using the LINCOS system. A more important reason for critical consideration of including proofs as a part of a message is the fact that detailed often lengthy proofs are as a rule *not* very illuminating. If a complicated logical expression, representing a fact, is written with a number of and's and or's, and perhaps nested lambda forms, the unwieldy nested verification is difficult to comprehend for humans (see e.g., the case of the Matrojshka's), so perhaps also for ETI. One might argue that this does not matter as long as an information processing machine (a computer) can handle such verifying expressions.

The handling of a fact would be in the first place asserting its type (the examples given indicate that these usually are already known) because the fact might be applicable to argument(s) or even occur in an application. The meaning of facts, the expressions designated by using the meta symbol=*in our notation*, is therefore not always of primary importance. So we conclude that the verifications themselves, important as they are in a general sense, not always are needed in messages for ETI. What *is* a necessary requirement, however, is the assurance that all facts included in annotations are verifiable (and have been verified).

So we might be restrictive in writing full-length verifications in actual message design. This book, however, does contain here and there lengthy proofs. The reason for that is that they do contain for human readers, valuable information on the way proof machinery works. At the same time it must be realized that the *dynamics* of verifications, the ways and means by which they are obtained, do contain valuable information, for instance in view of the nontrivial problem of the interpretation of the LINCOS system itself. Chapter 7 for instance contains rather lengthy proofs, hopefully informative for human readers. Separate we give ample attention to the dynamics of proof construction (within the conceptual bounds of the lambda calculus and the calculus of constructions) in Wittgenstein's Theatre (Chap. 8).

Chapter 5

Simple Facts

Intention

There is a multitude of kinds of facts in the reality of mankind. We do not suppose that each kind is expressible in the *Lingua Cosmica* discussed in the present treatise. This is because we are concerned with facts in a *logical* sense. Facts expressible in LINCOS consist as we have seen in examples in PART I nearly always of an implication or a sequence of implications leading from premises to a conclusion, written to the right of the rightmost implication token. The premises, i.e. one or more terms to the left of the token, contain necessary (not necessarily sufficient) information by which the conclusion can be verified. Other information may be needed; it should be available in and extractible from the *environment* (a logical database), sometimes in this treatise called the *stage*, see Chap. 7 and especially Chap. 8. The present chapter provides information on the way facts of a simple kind are introduced, verified and applied. We have already briefly mentioned and described some conventions around the concept of facts. Now we introduce in more detail and more systematically the topics of facts, lemma's or theorems and their verifications.

Elementary Facts

A fact to be verified is introduced by writing its name together with the type of that name, i.e. the premises and conclusion written as one term. In sequences of implications the rule is that brackets associate to the right and furthermore the conclusion, the rightmost term, can also be an implication. An introduced fact couples the name of the fact to its type, but they are of course not equal. Equality appears when the verification is achieved by a construction. Therefore the constructed verification of the fact is another expression of which the type is the same as the type of the fact.

Suppose that A, B and C are declared as type Prop. One of the simplest facts is

FACT f.5.1 : A $\rightarrow$ A.

The verification is $[h : A]h$ because the type of this expression is $A \rightarrow A$. In this verification $[h : A]$ takes care of the *premise* of the fact and the constructed h can be used in the *conclusion*. The entity h is written immediately behind the closing bracket of $[h : A]$ because $[h : A]$ is to be applied to h . So we write

$$f.5.1 = [h : A]h.$$

It can happen that not all of the premises of a fact are necessary for the conclusion, e.g.

$$\begin{aligned} \text{FACT } f.5.2 & : A \rightarrow B \rightarrow C \rightarrow A. \\ f.5.2 & = [h : A; * : B; * : C]h. \end{aligned}$$

The premises B and C are not needed for the conclusion. So we write $* : B$ and $* : C$ in the lambda abstraction, using the token $*$ for “it doesn’t matter”. In the next example some other constants are not needed.

$$\begin{aligned} \text{FACT } f.5.3 & : A \rightarrow B \rightarrow C \rightarrow (B \rightarrow C). \\ f.5.3 & = [* : A; * : B; h : C; * : B]h. \end{aligned}$$

It looks as though the conclusion is $(B \rightarrow C)$, but that is not the case. Under the convention that brackets associate to the right under implications, we have

$$[* : A; * : B; h : C; * : B]h : A \rightarrow B \rightarrow C \rightarrow B \rightarrow C$$

and this is the same as the type of fact $f.5.3$. Something similar is the case with

$$\begin{aligned} \text{FACT } f.5.4 & : (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C). \\ f.5.4 & = [h : A \rightarrow B \rightarrow C; h0 : A \rightarrow B; h1 : A](h h1) (h0 h1). \end{aligned}$$

This looks more complicated than it is. Because of the mentioned bracket convention for implication we have the equivalent

$$f.5.4 : (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C.$$

Therefore the conclusion is C , and a resident of C needs to be constructed.

Type checking is achieved by treating the lambda bindings as follows. Because $h : (A \rightarrow (B \rightarrow C))$ on account of the bracket convention, the first match scanning the lambda abstractions from left to right is $(h h1) : B \rightarrow C$. Then an entity of type B is needed, $(h0 h1) : B$. Finally $(h h1)$ applied to $(h0 h1)$ yields the required type C . Here the bracket convention for functional application (brackets associate to the left) is used. So $(h h1) (h0 h1) : C$. Note, however,

$$\begin{aligned} \text{FACT } f.5.5 & : ((A \rightarrow B) \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C). \\ f.5.5 & = [h : ((A \rightarrow B) \rightarrow C); h0 : (A \rightarrow B); * : A](h h0). \end{aligned}$$

Non-elementary Facts, Elimination Applied

Above facts are elementary because they do not involve inductively defined functions. We continue now with less elementary facts. Some results of the discourse in Part I are (in generalised form) the following hypotheses and verified facts.

```

HYPOTHESIS MP-gen : (x, y : Prop) (x /\ (x → y)) → y.
HYPOTHESIS MT-gen : (x, y : Prop) (~y /\ (x → y)) → ~x.
FACT proj1-gen : (x, y : Prop) (x /\ y) → x.
FACT proj2-gen : (x, y : Prop) (x /\ y) → y.
FACT f.5.6 : (x, y : Prop) (or x y) → (or y x).

```

The hypothesis Modus Ponens can be replaced by a fact:

```

FACT MP-gen : (x, y : Prop) (x /\ (x → y)) → y.

```

The verification is rather simple. Here it is in terms of ELIM, explained in Chap. 2.

```

MP-gen = [x,y : Prop; H : (x /\ (x → y))]
          (ELIM H [h1:x; h2:x→y]
           (h2 h1)).

```

The type of the fact MP-gen is $(x, y : \text{Prop})(x \wedge (x \rightarrow y)) \rightarrow y$ because x and y occur as the first local variables in the lambda form, and because, using $H : (x \wedge (x \rightarrow y))$

```

(ELIM H [h1:x; h2:x→y] (h2 h1)) : y.

```

The hypothesis for Modus Tollens can also be replaced by a fact:

```

FACT MT-gen : (x,y : Prop) (~y /\ (x → y)) → ~x.

```

Modus Tollens expresses: if not y is the case and if x implies y is the case, then not x is the case. In classic propositional logic MT can be justified as follows. Suppose that not y is the case. Further suppose $a1 : x$ (x is the case) and $a2 : (x \rightarrow y)$, i.e. x implies y is the case, then by Modus Ponens y is the case. Thus we have a contradiction and we must discard the supposition $a1 : x$. *Ergo*, x is the case must be discarded and the conclusion is that not x is the case. This argument cannot be used in constructive logic because there is no objection against $b1 : y$ (y is the case) and $b2 : \sim y$ (not y is the case) for some mutually distinct $b1$ and $b2$. These remarks suggest that the verification of MT-gen might be quite involved. There are certain subtleties to be explained.

```

MT-gen = [x,y : Prop; H : (~y /\ (x → y))]
          (ELIM H [h1: ~y; h2 : x→y; h3 : x]
           (h1 (h2 h3))) .

```

Under ELIM, $h1$ and $h2$ together select the induction hypothesis of the inductive $\wedge$. We take the liberty of using $h3$, an auxiliary extra local variable. So with $h1 = y \rightarrow \text{nil}$ (see Chap. 2 under *Negation*):

```
(h2 h3) : y
(h1 (h2 h3)) : nil.
```

Under ELIM the selectors h1 and h2 “disappear” (they are accounted for, in fact eliminated), but the variable h3 of type x does not. As a result we find

```
(ELIM H [h1: ~y; h2 : x→y; h3 : x] (h1 (h2 h3)) ) :
x → nil
or (ELIM H [h1: ~y; h2 : x→y; h3 : x] (h1 (h2 h3)) )
: ~x
```

leading to the desired result:

```
[x,y : Prop; H : (~y /\ (x → y))]
(ELIM H [h1: ~y; h2 : x→y; h3 : x] (h1 (h2 h3)) ) :
(~y /\ (x → y)) → ~x.
```

The use of ELIM as explained before might be considered to be outside pure type theory. One could improve on this by using another formalism instead, replacing this particular ELIM by the following expression associated with it:

```
HYPOTHESIS and-ind : (x',y',P : Prop) (x'→y'→P)→
(x' /\ y'→P) .
```

Here P is the entity in Prop for which a resident must be constructed. As mentioned before, $\wedge : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$. The suffix “ind” seems to suggest that this is an inductive definition, but it is of course not. The hypothesis is a so-called *inductive form* (as used in the Project Coq) discussed in more detail in Chap. 13.

For Modus Ponens one obtains then with $P = y$ and remembering to read x for x' and $(x \rightarrow y)$ for y' , the following:

```
MP-gen = [x,y : Prop; h : (x /\ (x → y))]
(and-ind x x→y y [h1:x; h2: x→y] (h2 h1) h) .
```

Annotation 1

The abovementioned and-ind in MP-gen requires three arguments and then must be applied to two arguments in order to find a resident of y. The first three are x, $x \rightarrow y$, and y of type Prop. Because (h2 h1): y, the fourth argument is

```
[h1:x; h2: x→y] (h2 h1) : x→ (x→y) → y.
```

Finally the fifth is $h : (x \wedge (x \rightarrow y))$. *Summa summarum*

```
(and-ind x x→y y [h1:x; h2: x→y] (h2 h1)) h) : y.
end Annotation 1
```

In pure form we have for Modus Tollens with $P = \sim x$ the following expression.

```
MT-gen = [x,y : Prop; h : (~y /\ (x → y))]  
(and-ind ~y x→y ~x [h1:~y; h2: x→y; h3:x] (h1 (h2 h3)) h) .
```

Annotation 2

Regarding the arguments of and-ind in MT-gen one observes the following. The first three are $\sim y$, $x \rightarrow y$, and $\sim x$ (corresponding to P) of type Prop. The fourth is

```
[h1:~y; h2: x→y; h3:x] (h1 (h2 h3)) : ~y→(x→y)→~x
```

as we have seen before.

Finally the fifth is $h : (\sim y \wedge (x \rightarrow y))$. *Summa summarum*

```
(and_ind x x→y y [h1:x; h2: x→y; h3: x] (h1 (h2 h3))  
h) : ~x. end Annotation 2
```

Existence Revisited

In Chap. 2 we discussed the existence function Ex and derivations of facts using elimination, more in particular ELIM. The latter function might be considered to be outside of pure type theory. One could justify its application by using the following hypothesis instead (see the Coq project).

```
HYPOTHESIS Ex-ind : (X : Set; P : X→Prop; R : Prop)  
((x : X) (P x) → R) → (Ex X P) → R.
```

Ex -ind (also an inductive form) can be considered to be associated with Ex .

The structure of Ex -ind is similar to that of and-ind, discussed in the present chapter. Like in that case, here the arguments of Ex -ind are those of Ex augmented with an extra argument R of type Prop. The induction hypothesis, likewise extended to $((x : X)(P x) \rightarrow R)$ is present.

Further Ex itself occurs in the hypothesis; for its type we have $Ex : (X : Set)(X \rightarrow Prop) \rightarrow Prop$, as in Chap. 2. R is the entity in Prop for which a resident must be constructed in proof mode.

In order to illustrate the use of Ex -ind we shall now verify separately *The Drinker's Paradox* discussed already in Chap. 2. We assume given declarations of

$S : Set$ (the bar), $s : S$ (there is a drinker in the bar) and $Q : S \rightarrow Prop$. Moreover it will appear that we need a curious extra hypothesis.

```
HYPOTHESIS cs : (x : S) (Q x) → (y : S) (Q y) .
```

Then the paradox is expressed as a fact:

```
FACT DP : (Ex S Q) → (x:S) (Q x) .
```

In order to verify this fact *via* Ex-ind, we must construct a resident of $(x:S)(Q\ x)$ and so we must use $R = (x:S)(Q\ x) : \text{Prop}$. Ex-ind requires 5 arguments. The first three are obtained by substituting as follows:

S for X , Q for P , $(x:S)(Q\ x)$ for R .

The fourth is then of course the curious cs , and the fifth $h : (Ex\ S\ Q)$ is obtained in the customary way (from the premise of DP). Therefore

$(\text{Ex-ind}\ S\ Q\ (x:S)(Q\ x)\ cs\ h) : R$.

Here is the final construction.

$DP = [h : (Ex\ S\ Q)] (\text{Ex-ind}\ S\ Q\ (x:S)(Q\ x)\ cs\ h) .$

$DP : (Ex\ S\ Q) \rightarrow (x:S)(Q\ x) .$

This example shows that using ELIM, as done in Chap. 2, yields a much simpler expression.

$DP = [h : (Ex\ S\ Q)] (\text{ELIM}\ h\ [y : (x:S)(Q\ x)]y) .$

We did not need the curious hypothesis explicitly in that case because we accepted the ELIM expression above. We usually employ ELIM in this book.

Chapter 6

Compounds

Intention

Facts less simple than the examples discussed in Chap. 5 are those containing several inductively defined functions. In the present chapter we consider in some detail facts over expressions containing conjunctions and disjunctions. They are seemingly simple, but they are far from trivial since the logical connectives are inductively defined. As a result the verifications of these kinds of facts are much more involved than in the easy cases.

Commutativity, Transitivity and Distributivity

Let A, B and C be declared as entities of type Prop. We have remarked in a note in a previous chapter how the commutativity of conjunction ($\wedge$) can be shown. We have seen

FACT f.6.1 : (A $\wedge$ B) $\rightarrow$ (B $\wedge$ A) .

with the verification

f.6.1 = [H : A $\wedge$ B] (ELIM H [h1 : A; h2 : B] (Conj B A h2 h1))

where from the inductive definition of $\wedge$

Conj : (X, Y : Prop) X $\rightarrow$ Y $\rightarrow$ (X $\wedge$ Y) .

In above formulation ELIM is given the local variable H (lambda bound) as an extra argument. This is not imperative in this case as there is only one inductive function “and” to be eliminated. When more than one is to be eliminated, it is necessary to use the names of the local variables in the lambda abstractions to distinguish them from each other in the ELIM parts. It can also happen that an application of a local variable is needed as in Chap. 5.

Verifications can be rather complicated nested structures. We expect commutative, transitive and distributive relations to hold over compounds with the operators $\wedge$ and $\rightarrow$. For example

```
FACT f.6.2 : ((A /\ B) /\ C) → ( A /\ C) .
f.6.2 = [H : ((A /\ B) /\ C)]
        (ELIM H [h1 : A /\ B; h2 : C]
         (ELIM h1 [h11 : A; * : B] (Conj A C h11 h2) ) ) .
```

Here we have two “levels” of eliminations. In the next example there are even three levels.

```
FACT f.6.3 : (A /\ B) /\ (B /\ C) → (A /\ B) .
f.6.3 = [H : (A /\ B) /\ (B /\ C)]
        (ELIM H [h1 : A /\ B; h2 : B /\ C]
         (ELIM h1 [h11 : A; h12 : B]
          (ELIM h2 [* : B; h22 : C] (Conj A B h11 h12)
           ) ) ) .
```

Modus Tollens

We have seen in Chap. 5 how the law Modus Tollens can be proved. Starting from the idea used in classical reasoning, the rather unusual constructive verification is obtained there. The law, writing simply MT because the generalization “for all x and y ” is implied by $(x, y : \text{Prop})$, stated as a hypothesis reads as follows:

```
HYPOTHESIS MT : (x, y : Prop) (~y /\ (x → y)) → ~x .
```

Rules such as the double negation (DN), and the De Morgan’s Laws (DM) can best be introduced as hypotheses in general forms:

```
HYPOTHESIS DN1 : (x : Prop) (~ ~ x → x) .
HYPOTHESIS DN2 : (x : Prop) (x → ~ ~ x) .

HYPOTHESIS DM1 : (x, y : Prop) (~(x /\ y)) → (~x \/ ~y) .
HYPOTHESIS DM2 : (x, y : Prop) (~x \/ ~y) → ~(x /\ y) .
HYPOTHESIS DM3 : (x, y : Prop) (~(x \/ y)) → (~x /\ ~y) .
HYPOTHESIS DM4 : (x, y : Prop) (~x /\ ~y) → ~(x \/ y) .
```

Logic in Sentences

Now we proceed with examples of compound facts occurring as annotations to sentences—showing LINCOS at work! At this stage we do give the verifications, but remark that we do not take the position that in LINCOS annotations to text

should always be accompanied by formal proofs. Whether or not to include the verifications depends on their informative value—see also the discussions on the workings of the proof machinery in Chap. 8.

Consider

Socrates is a human and all humans are mortal. Therefore Socrates is mortal.

In Chap. 7 we explain that this is not a very complex sentence and is not an Aristotelian syllogism, since the so-called *singular* (Socrates) occurs. We write anyhow for this stand-alone sentence an annotation. Let the environment be

```

CONSTANT Socrates : Set.
CONSTANT human : Set → Prop.
CONSTANT mortal : Set → Prop.

FACT f.6.4 :
  (human Socrates) /\ (x:Set) ((human x) → (mortal x)) →
  (mortal Socrates).

f.6.4 = [H : (human Socrates) /\ (x:Set) ((human x) →
  (mortal x))]
      (ELIM H [h1 : (human Socrates);
        h2 : (x:Set) ((human x) → (mortal x))]
        (h2 Socrates h1)).

```

A little more complicated is

John is a human and all humans like leisure and like being healthy.

Therefore John likes leisure and being healthy.

Use the stage (i.e. environment)

```

CONSTANT John : Set.
CONSTANT human : Set → Prop.
CONSTANTS like-leisure, like-healthy : Set → Prop.

FACT f.6.5 :
  (human John) /\ (x:Set) ((like-leisure x) /\ (like-
  healthy x)) →
  (like-leisure John) /\ (like-healthy John).

f.6.5 = [H : (human John) /\ (x:Set) ((like-leisure x)
  /\ (like-healthy x))]
      (ELIM H [h1 : (human John);
        h2 : (x:Set) ((like-leisure x) /\
          (like-healthy x))]
        (ELIM (h2 John) [h3 : (like-leisure
          John);
            [h4 : (like-healthy
              John)]
          (Conj (like-leisure John) (like-healthy
            John) h3 h4)) ) ).

```

Note the use of (h2 John) under the second ELIM. This enables the proof mechanism to “pick up” the induction hypothesis of $(\text{like-leisure } x) \wedge (\text{like-healthy } x)$ in this case. See Chap. 8.

Next consider

Mary is a woman and Mary likes pancakes and all women like to cook.

Therefore somebody likes pancakes and likes to cook.

The stage is

```

CONSTANT Mary : Set.
CONSTANT woman : Set → Prop.
CONSTANT like-pancake : Set → Prop.
CONSTANT like-to-cook : Set → Prop.

FACT f.6.6 :
  (woman Mary) /\ (like-pancake Mary) /\ (x : Set)
  ((woman x) → (like-to-cook x)) →
  (Ex Set like-pancake) /\ (Ex Set like-to-cook).

```

Here we need to use Ex over Type (Chap. 2 for the case Ex over Set).

```

INDUCTIVE Ex [X : Type; P : X → Prop] : Prop :=
  Ex-intro : (x : X) (P x) → (Ex X P).

```

```

f.6.6 =
  [H : (woman Mary) /\ (like-pancake Mary) /\
  (x : Set) ((woman x) → (like-to-cook x))]
  (ELIM H [h1 : (woman Mary);
           h2 : (like-pancake Mary) /\
           (x : Set) ((woman x) → (like-to-cook x))]
  (ELIM h2 [h3 : (like-pancake Mary) ;
            h4 : (x : Set) ((woman x) → (like-to-cook
            x))]
  (Conj (Ex Set like-pancake) (Ex Set like-to-cook)
        (Ex-intro Set like-pancake h3)
        (Ex-intro Set like-to-cook ((h4 Mary) h1)) ) ) ).

```

This rather heavy formalism results from the form of the conclusion, where the two requirements $(\text{Ex Set like-pancake})$ and $(\text{Ex Set like-to-cook})$ are kept apart from one another. In Chap. 7 another existence function Ex2-and is defined over Type and the conjunction over two properties P_1 and P_2 simultaneously. It can be used with like-pancake and like-to-cook in above case as well.

The next example is slightly different from the former.

Some humans like pancakes and all humans that like pancakes like wine.

Therefore somebody likes wine.

In this case we use the stage

```

CONSTANT human : Set.
CONSTANT like-pancake : human → Prop.

```

```

CONSTANT like-wine : human → Prop.

FACT f.6.7 :
  (Ex human like-pancake) /\ ((x : human)(like-pancake
x) → (like-wine x)) →
  (Ex human like-wine).

```

As the conclusion does not specify who likes wine, we have not declared a resident of human. Consequently in the verification we need the interpretation of the induction hypothesis of the existence function Ex, which does not refer to such a resident (see Chap. 2).

```

f.6.7 =
[H : (Ex human like-pancake) /\ ((x : human)(like-
pancake x) → (like-wine x))]
(ELIM H [h1 : (Ex human like-pancake);
        h2 : ((x : human)(like-pancake x) → (like-
wine x))])
(ELIM h1 [x : human; h3 : (like-pancake x)]
(Ex-intro human like-wine x (h2 x h3)) ) ).

```

The next example involves several times “some”.

Some humans are long-haired
and some long-haired humans are philosophers
and all philosophers are eccentric.
Therefore some long-haired humans are eccentric.
The stage is

```

CONSTANT human : Set.
DEFINE long-haired-human := human.
CONSTANT is-long-haired : long-haired-human → Prop.
CONSTANT is-philosopher : human → Prop.
CONSTANT is-eccentric : human → Prop.

```

In this environment we first give the type of human and then state that there is no difference between human and long-haired-human. This does *not* imply that all humans are long-haired! The type definition of long-haired-human is introduced here in order to be able to express “some long-haired humans are philosophers”. With this stage the fact expressing the premises and conclusion (“Therefore some long-haired humans are eccentric”) takes the following form

```

FACT f.6.8 :
  (Ex human is-long-haired) /\ (Ex long-haired-human
is-philosopher) /\
  ((x : human)((is-philosopher x) → ((is-eccentric x))
→
  (Ex long-haired-human is-eccentric)

```

with the verification

```

f.6.8 =
[H : (Ex human is-long-haired) /\ (Ex long-haired-
human is-philosopher) /\
((x : human)((is-philosopher x) → ((is-eccentric x)))
(ELIM H [h1 : (Ex human is-long-haired);
        h2 : (Ex long-haired-human is-philosopher) /\
              ((x : human)((is-philosopher x) →
              ((is-eccentric x)))
(ELIM h1 [x : human; h3 : (is-long-haired x)]
(ELIM h2 [h4 : (Ex long-haired-human is-philosopher);
          h5 : (x1 : human)((is-philosopher x1) →
              ((is-eccentric x1))]
(ELIM h4 [y : long-haired-human; h6 : (is-philosopher
y)]
(Ex-intro long-haired-human is-eccentric y (h5 x h6))
) ) ) ).

```

We have remarked before that verifications such as these shown above must exist (otherwise one cannot state a fact), but that the information content is questionable as far as usefulness is concerned. We return to this matter in Wittgenstein's theatre, Chap. 8.

Complexity of the Stage

In concluding the present chapter we draw the attention to some important aspects of writing LINCOS annotations, different from earlier examples. Note that each of the examples shown requires a special stage (more generally environment) as well as its specific treatment. The text is *mirrored* in the stage well as the formulated fact. That means that the text to be annotated could be judged in terms of the complexity of the stage and the fact(s). A complexity measure might be the number of lines combined with a weighted mean of the constructs employed. If a given sequence of sentences needs a complex stage or set of facts as annotations in LINCOS, one might consider designing another, equivalent text, imposing less complexity.

In the field of interstellar message construction, this might be an important matter. Do we wish to construct messages perspicuous to ourselves or to those at the receiving end—or perhaps to satisfy both requirements? In the first case we would use grammatical constructs easy to understand for humans. But that does not necessarily mean that the complexity mentioned is minimal—experience indicates that in general it is not. On the other hand if at the receiving end information processing artefacts using computer programs are employed to do the interpretation, the complexity of stages and facts are less important. We shall not go into this matter in this book, explaining instead in some detail how LINCOS is used. As an aid for this purpose many examples are worked out.

Chapter 7

Aristotelian Theatre

A deduction is speech (logos) in which, certain things having been supposed, something different from those supposed results of necessity because of their being so.

Aristotle (384–322, B.C.) in *Prior Analytics*, transl. A.J. Jenkinson

Intention, Logic of Sentences

Terms in the formalism of the calculus of constructions dealt with so far have mostly simple structures. They have not a posteriori been formulated in natural language, although it could easily have been done. With new LINCOS in mind, we take in the present chapter an opposite view and look at simple linguistic structures, i.e. sentences in natural language, and their logical contents. This approach is taken in preparation of the discussion of the more general case of conglomerates of sentences, i.e. texts, in Parts III and IV. The logic in the present chapter and when discussing texts will of course be expressed in the conventions of the lambda calculus and more in particular of the calculus of constructions explained in the earlier chapters. In that way we have here once again examples of LINCOS applications.

In the present chapter sequences of words forming sentences, in fact propositions, will be in the Aristotelian theatre. They are in fact of the special kind of logic introduced more than 2000 years ago by the Greek philosopher Aristotle (384–322 B.C.) in his book *Organon*—see the famous fresco in Rome painted by Rafaël in 1509–1511, depicting Plato (428–348 B.C.) and him in the “Athenian School of Philosophy” (also “Disputà”).

More in particular sentences contain (are concerned with) subjects and predicates. A subject usually designates a collection of objects, e.g. *Dutchmen, rabbits, shrubs, etc.* A predicate always designates something universal, e.g. *poor, animal, white, etc.* In addition to this propositions are non-singular, i.e. they are not concerned with individuals.

Note: a useful quick review of Aristotelian syllogisms is contained in the chapter Postscriptum, Appendix E. See also (Ollongren [2011](#)).

Simple Cases

First we suppose that sentences have a single subject *S* predicated by a single predicate *P*. Furthermore such a sentence must either affirm or deny the predicate. Because there are two kinds of subject quantifications (“all” and “some”), there are four kinds of *Aristotelian* predications.

Universal	- all <i>S</i> is <i>P</i>	- no <i>S</i> is <i>P</i>
Particular	- some <i>S</i> is <i>P</i>	- not every <i>S</i> is <i>P</i>

In this section we show how instances of these kinds can be represented in LINCOS. We consider the following four examples, i.e. four instances of the occurring predications (human, animal, having a beard, male) of only one subject. In honour of Aristotle we choose Greeks as the subject in instances 1–4.

1. All Greeks are human
2. No Greek is animal
3. Some Greeks have beards
4. Not every Greek is male

In these examples there is only one subject: Greeks. Before we can show representations (i.e. in LINCOS) of these sentences, the stage must be set. There are several ways to do so. Let us begin with the partial stage

CONSTANT Greek : Set.

CONSTANTS is-human, is-animal : Greek $\rightarrow$ Prop.

Case 1 (an Aristotelian A term, see next section) is represented by

HYPOTHESIS h.7.1 : $(x:\text{Greek}) (\text{is-human } x)$.

Case 2, reformulated as “all Greeks are not animal” (an Aristotelian E term, see next section), is represented by

HYPOTHESIS h.7.2 : $(x:\text{Greek}) \sim(\text{is-animal } x)$.

Note that is-human and is-animal have the same type. Thus if $A : \text{Greek}$, $(\text{is-human } A) : \text{Prop}$ and $(\text{is-animal } A) : \text{Prop}$. By the way $\sim(\text{is-human } A) : \text{Prop}$ and $\sim(\text{is-animal } A) : \text{Prop}$ as well. This shows that the property that an entity is of type Prop, usually needs interpretation. In the case of the Greeks we must see to it that humans and animals are kept apart in some way.

Now we introduce a representation *G* of Greeks in the real world. This is in order to be able to treat cases 3 and 4 constructively. So here is *G*

CONSTANT *G* : Greek.

Then for each Greek the following facts

FACT f.7.1.1 : $(\text{is-human } G)$.

FACT f.7.2.1 : $\sim(\text{is-animal } G)$.

are verified by

`f.7.1.1 = (h.7.1 G) .`

`f.7.2.1 = (h.7.2 G) .`

For cases 3 and 4 the stage must be extended with

`CONSTANT is-bearded : Greek → Prop.`

`CONSTANT is-not-male : Greek → Prop.`

Thus `(is-bearded G) : Prop`.

Case 3, reformulated as “There exists a Greek with a beard”, (an Aristotelian I term, see next section), is represented by

`FACT f.7.3.1 : (Ex Greek is-bearded) → (is-bearded G) .`

The inductive definition of `Ex` is in Chap. 2.

`INDUCTIVE Ex [X : Set; P : X → Prop] : Prop :=`
`Ex-intro: (x : X) (P x) → (Ex X P) .`

The fact `f.7.3.1` does not exclude the possibility that all Greeks in the real world have beards. Cf. *The Drinker’s Paradox* discussed in Chaps. 2 and 5. The fact is non-trivial because a resident of `(is-bearded G)` must be constructed in order to prove it. The verification (eliminating the induction hypothesis from the definition of `Ex`) is

`f.7.3.1 =`
`[h : (Ex Greek is -bearded)]`
`(ELIM h [h1 : (x:Greek) (is-bearded x)] (h G) ) .`

Case 4, reformulated as “there exists a Greek not a male”, (an Aristotelian O term, see next section), is represented by

`FACT f.7.4.1 :`
`(Ex Greek is-not-male) → (is-not-male G) .`

This does not exclude that no Greeks in the real world are male. That would for instance be the case if the real world considered (the universe of discourse) consists of some congregation of women. The verification of the fact `f.7.4.1` is

`f.7.4.1 =`
`[h : (Ex Greek is-not-male)]`
`(ELIM h [h1 : (x:Greek) (is-not-male x)] (h G) ) .`

Note the special way in which negation is treated in case 4. We have declared `is-not-male`. Had we declared `is-male` of type `Greek → Prop`, then we would have been forced to use `~(is-male G)` of type `Prop` as the second argument of `Ex`. However, the first is `Greek` (of type `Set` as required by the definition of `Ex`), so we would have a clashing of types. Another remark: in above facts the environmental information is of prime importance. See Chap. 8 for further details.

Aristotelian Conversions

In Aristotelian logic syllogisms (propositions) have the following form:

major term and *minor* term imply *conclusion* term.

We have seen (Cases 1–4) that there are four mutually different terms in this logic. They are called A, E, I and O terms. Two of them (A and I) are affirmative (*affirmo*), and two (E and O) are negative (*nego*). A and E terms are of *universal* kinds, I and O terms are of *particular* kinds. Subjects occurring in terms are again represented by S, predicates by P (*cf.* above section **Simple cases**). Slightly different from the previous section the terms now are designated by

Asp all S are P
 Esp no S is P
 Isp some S is P
 Osp some S are not P.

In the following we show how instances of these kinds can be represented in LINCOS.

Remark: there are no *singular* terms in this logic. The structure of the proposition (instance 5)

5. all humans are mortal, Socrates is a human, therefore Socrates is mortal, is: major term, minor term, conclusion term. The proposition is, however, not in Aristotelian logic, because the singular “Socrates is a human” occurs in it. In fact Aristotelian logic is unable to deal with individuals as subjects. In scholarly commentaries on Aristotelian logic this has been considered to be a weakness. Fortunately the logic contents of sentences like the one here can be expressed in the LINCOS system, even though individuals must be represented in a special way (so that their individuality is guaranteed, see Chap. 14). Note that in previous sections we have avoided introducing individuals by using G as a representation of Greeks.

In the simple cases treated in the previous section we had one subject and different predications. Now we look at the case of various terms consisting of one subject and one predication under various combinations of “all” and “some” quantifications together with negations. More in particular we consider in this section *conversion rules* for transforming one term into another with invariant subject and predication—themselves considered to be interchangeable (i.e. a subject can be used as a predication and the other way around). *Simple* conversions are those for which the kind of the term is also invariant, *per accidens* are those for which the kind changes. The complete Aristotelian set of conversions is

Asp --- > Ips, all S are P → some P is S
 (per accidens)
 Esp --- > Eps, no S is P → no P is S (simple)
 Eps --- > Ops, no P is S → some P are not S
 (per accidens)
 Isp --- > Ips, some S is P → some P is S. (simple)

The validity of these conversions is easily understood in set-theoretic terms. Let S and P be represented by sets.

- Let S be contained in (is a subset of) P .
- Then $\text{Asp} \dashv\dashv \rightarrow \text{Ips}$.
- Let S and P be disjoint (empty intersection).
- Then $\text{Esp} \dashv\dashv \rightarrow \text{Eps}$, but also $\text{Eps} \dashv\dashv \rightarrow \text{Ops}$.
- Let the intersection of S and P be non-empty.
- Then $\text{Isp} \dashv\dashv \rightarrow \text{Ips}$.

Logical Implications

In the present section we discuss these conversions regarded as logical implications. The subjects S and the predicates P are declared (note that they are given the same type) with

HYPOTHESIS $S, P : \text{Set} \rightarrow \text{Prop}$.

In addition we assume that Set is non-empty with the introductory abstraction

CONSTANT $s : \text{Set}$

The A, E, I and O terms are modeled by

Asp all S are P

INDUCTIVE $A[x:\text{Set}; S, P : \text{Set} \rightarrow \text{Prop}] : \text{Prop} :=$
 $A\text{-intro} : (S\ x) \rightarrow (P\ x) \rightarrow (A\ x\ S\ P).$

Esp no S is P

INDUCTIVE $E[x:\text{Set}; S, P : \text{Set} \rightarrow \text{Prop}] : \text{Prop} :=$
 $E\text{-intro} : ((S\ x) \rightarrow \sim(P\ x)) \rightarrow (E\ x\ S\ P).$

Isp some S is P

INDUCTIVE $I[x:\text{Set}; S, P : \text{Set} \rightarrow \text{Prop}] : \text{Prop} :=$
 $I\text{-intro} : (S\ x) /\ (P\ x) \rightarrow (I\ x\ S\ P).$

Osp some S are not P .

INDUCTIVE $O[x:\text{Set}; S, P : \text{Set} \rightarrow \text{Prop}] : \text{Prop} :=$
 $O\text{-intro} : (S\ x) /\ \sim(P\ x) \rightarrow (O\ x\ S\ P).$

In this way we need not give separate definitions for Asp and Aps , etc.

In order to model the conversion rules in a logic sense, we interpret the arrow $\dashv\dashv \rightarrow$ as an implication in the sense of the calculus of constructions. Verifiable conversions then become facts.

Case $\text{Asp} \dashv\dashv \rightarrow \text{Ips}$.

FACT $f.7.5.1 : (A\ s\ S\ P) \rightarrow (I\ s\ P\ S).$

Using

```
I-intro : (x : Set; P, S : Set → Prop) (P x) /\ (S x) →
(I x P S).
(I-intro s) : (P, S : Set → Prop) (P s) /\ (S s) → (I s
P S).
```

and remembering from Chap. 2

```
Conj : (X, Y : Prop) X → Y → (X /\ Y).
```

we find

```
f.7.5.1 = [* : (A s S P)]
(ELIM-A [h1 : (S s); h2 : (P s)](I-intro s P S (Conj
(P s) (S s) h2 h1)) ).
```

ELIM-A is used here to eliminate the induction hypotheses in the definition of A. In this way the local variables h1 and h2 make their appearance on the scene.

Note. An alternative definition of I is

```
INDUCTIVE I' [x:Set; S, P : Set → Prop] : Prop :=
  I-intro' : (Ex2-and Set S P) → (I' x S P).
```

This shows that the induction condition $(S x) \wedge (P x)$ need not be explicit. It can be embedded. The rather unconventional induction hypothesis $(\text{Ex2-and Set } S P)$ can be used for “hiding” the condition $(S x) \wedge (P x)$ in an existence function (called Ex2), itself a modification of the existence function Ex. The declaration of the existence function Ex of *one* property P1 of an entity x of type X given in Chap. 2 is reproduced earlier in the present chapter.

Here we introduce a novelty with Ex2-and. This is an extension of Ex to the case of *two* properties P1 and P2 of an entity x of type X, but then such that

$(P1 x) \wedge (P2 x)$ is the existence condition.

```
INDUCTIVE Ex2-and [X:Type; P1, P2: X → Prop] : Prop :=
  Ex2-and-intro : (x:X) ((P1 x) /\ (P2
x)) → (Ex2-and X P1 P2).
```

```
Ex2-and-intro : (X:Type; P1, P2 : X → Prop; x:X)
((P1 x) /\ (P2 x))
→ (Ex2 X P1 P2).
```

Using I', the fact f.7.5.1 and its verification change to

```
FACT f.7.5.1' : (A s S P) → (I' s P S).
```

```
f.7.5.1' = [H : (A s S P)]
(ELIM [h1 : (S s); h2 : (P s)]
(I-intro' s P S (Ex2-and-Intro Set P S s
(Conj (P s) (S s) h2 h1)) ).
```

because

```
I-intro' : (x : Set; P, S : Set → Prop) (Ex2-and Set
P S) → (I' s P S).
(I-intro' s P S) : (Ex2-and Set P S) → (I' s P S).
(Ex2-and-intro Set P S s) : ((P s) /\ (S s)) → (Ex2-
and Set P S).
```

This looks rather formidable because of the nesting. But the nesting is in forward sense with no backtracking. The building stones are simple. ELIM is used here again to get the local variables $h1$ and $h2$ onto the scene—and they are used in the construction of a resident of $(I' s P S)$. That entity is constructed by applying $I\text{-intro}'$ to four arguments—the last one is a resident of $(\text{Ex2-and Set } P S)$. Ex2-and requires a resident of $(P s) \wedge (S s)$, so we use the selector Ex2-and-intro supplied with five arguments. The fifth is Conj applied to four arguments to construct a resident of in the inductive definition of $\wedge$; they are

$(P s)$, $(S s)$, $h2$ and $h1$ as in the case of f.7.5.1.

Case Esp --- > Eps.

```
FACT f.7.5.2 : (E s S P) → (E s P S).
```

The definition of E for the case $(E s P S)$ means for the selector

```
E-intro : (x : Set; P, S : Set → Prop) ((P x) → ~(S
x)) → (E x P S).
(E-intro s (P s) (S s)) : ((P s) → ~(S s)) → (E s P S).
```

In formulating the verification we shall need

```
HYPOTHESIS h.7.5.2.1 : ((S s) → ~(P s)) → ((P s) →
~(S s)).
```

This implication cannot be proved constructively as a fact because a needed assumption (that $(P s)$ is the case) is not available. Using the hypothesis the verification of f.7.5.2 is easy.

```
f.7.5.2 =
[H : (E s S P)
(ELIM H [k : ((S s) → ~(P s))](E-intro s (P s) (S s)
(h.7.5.2.1 k)) )].
```

Case Eps --- > Ops.

```
FACT f.7.5.3 : (E s P S) → (O s P S).
f.7.5.3 = [H : (E s P S)]
(ELIM H [h1 : (P s); h2 : ~(S s)](O-intro s P S (Conj
(P s) ~(S s) h1 h2)) ).
```

The structure of this verification is similar to that of fact f.7.5.1.

Case Isp --- > Ips.

```
FACT f.7.5.4 : (I s S P) → (I s P S).
```

Because the induction hypothesis of I is $(S\ x) \wedge (P\ x)$ this fact should have a simple verification. Here it is.

```
f.7.5.4 = [H : (I s S P)]
(ELIM H [h1 : (P s); h2 : (S s)], (I-intro s P S
(Conj (P s) (S s) h1 h2)) ).
```

Even though the facts written are consistent with the conventions of LINCOS, above treatment is strictly speaking outside the system. Again this is because we have not used linguistic sentences to be annotated or modeled.

Figures

A *figure* of a syllogism is a sequence of propositions. In such a sequence first the major term is written, followed by the minor one and the conclusion is the last term from left to right. In Aristotelian logic figures are called *moods*. There are several moods possible. In Aristotle's writings the following four sequences of propositions are considered to be *perfect* because they are close to natural reasoning. Medieval logicians have coined the moods with the names written below

A A A
 E A E *Celarent*
 A I I *Darii*
 E I O *Ferio*.

An example of the universal figure A A A is
 all S1 are P and all S2 are S1, then all S2 are P.

The major term is "all S1 is P". The minor term is "all S2 are S1" and the conclusion term is "all S2 are P". S2 occurs in the major and in the minor term as well. Such a term is called a *middle* term and it acts as a bridge between the major term and the conclusion.

Because we are discussing the LINCOS system in this book, we shall not provide formal proofs of conclusions on the basis of abstract major and minor terms (the premises) as in the example above. Instead we take an instance of the figure considered. Since the instance keeps the pattern of the figure, the verification of it is a blue print for the general verification. In above case we use an example of the figure A A A as instance 6 as follows

6. if all men are mortal and all Greeks are men, then all Greeks are mortal.

Note: the form of this sentence is according to the pattern "if ... then ...". It will therefore be no surprise that the LINCOS fact expressing case 6 will contain $\rightarrow$, the implication token. Before formulating the conclusion as a fact we set the stage by

```
CONSTANTS is-man, is-Greek, is-mortal : Set → Prop.
HYPOTHESIS h.7.6.1 : (x : Set) (is-man x) → (is-mortal x).
```

HYPOTHESIS h.7.6.2 : $(x : \text{Set}) (\text{is-Greek } x) \rightarrow (\text{is-man } x).$

In this setting we have the fact

FACT f.7.6.3 : $(x : \text{Set}) (\text{is-Greek } x) \rightarrow (\text{is-mortal } x).$

The verification is

f.7.6.3 = $[x : \text{Set}; h : (\text{is-Greek } x)] (h.7.6.1 \ x \ (h.7.6.2 \ x \ h)).$

An example of the universal figure E A E is

- all S1 are not P1 and all S2 are S1, then all S2 are not P1.

The major term is “all S1 are not P1”, the minor term is “all S2 are S1”, S1 is the middle term and the conclusion term is “S2 are not S1”. Here the subject S1 has turned into a predication. The following is an instance of this case (instance 7)

7. if all Greeks are not animals and all Athenians are Greeks, then all Athenians are not animals.

CONSTANTS is-Greek, is-Athenian, is-animal : Set : Prop.

HYPOTHESIS h.7.7.1 : $(x : \text{Set}) (\text{is-Greek } x) \rightarrow \sim(\text{is-animal } x).$

HYPOTHESIS h.7.7.2 : $(x : \text{Set}) (\text{is-Athenian } x) \rightarrow (\text{is-Greek } x).$

FACT f.7.7.3 : $(x : \text{Set}) (\text{is-Athenian } x) \rightarrow \sim(\text{is-animal } x).$

f.7.7.3 = $[x : \text{Set}; h : (\text{is-Athenian } x)] (h.7.7.1 \ x \ (h.7.7.2 \ x \ h)).$

The proof of the LINCOS fact f.7.7.3 is a verification of an instance of *Celarent*.

An example of the universal figure A I I is

- all S1 are P1 and there are S1 which are P2, then there are P1 which are P2.

The major term is “all S1 are P1”, the minor term is “some S1 are P2”, the middle term is S1 and the conclusion term is “there are P1 which are P2”. Here the predication P1 has turned into a subject. The following is an instance of this case (instance 8)

8. if all Greeks are human and there exist Greeks which are females, then there exist humans which are females.

We need the existence functions over one and two predications. The inductive definition of Ex over one predication P is in Chap. 2.

INDUCTIVE Ex $[X : \text{Set}; P : X \rightarrow \text{Prop}] : \text{Prop} :=$
 Ex-intro: $(x : X) (P \ x) \rightarrow (\text{Ex } X \ P).$

```
Ex-intro : (X : Set; P : X → Prop; x : X) (P x) → (Ex
X P).
```

In addition we shall need the existence function over predications P1 and P2:

```
INDUCTIVE Ex2-and [X : Set; P1, P2: X → Prop] : Prop :=
  Ex2-and-intro : (x : X) ((P1 x) /\ (P2 x)) →
    (Ex2-and X P1 P2).
Ex2-and-intro : (X : Set; P1, P2 : X → Prop; x : X)
  ((P1 x) /\ (P2
x)) → (Ex2-and
X P1 P2).
```

Here is the formalization of instance 8.

```
CONSTANTS is-Greek, is-human, is-female : Set : Prop.
HYPOTHESIS h.7.8.1 : (x : Set)(is-Greek x) →
(is-human x).
HYPOTHESIS h.7.8.2 : (Ex2-and Set is-Greek
is-female).
FACT f.7.8.3 : (x:Set)((is-human x) /\ (Ex x
is-female)) →
  (Ex2-and x is-human is-female)
```

with the verification

```
f.7.8.3 =
[x : Set; H : (is-human x) /\ (Ex x is-female)]
(ELIM H [h1 : (is-human x); h2 : (Ex x is-female)],
  (ELIM h2 [h21 : (is-female x)],
    (Ex2-and-intro x is-human is-female x
      (Conj (is-human x) (is-female
x) h1 h21)) ) ).
```

Here we see once again how ELIM is used. First, using the induction hypothesis of the conjunction $\wedge$, h1 and h2 with their types appear on the scene. h1 can immediately be used, h2 not. So, at a lower level, h2 is eliminated and h21 with its type appears. For the conclusion not h2 but h1 and h21 are needed.

The proof of the LINCOS fact f.7.8.3 is a verification of an instance of *Darii*.

An example of the figure E I O is

- no S1 are P1 and some P1 are S2 while no S2 are S1, then there are no S1 which are S2.

The major term is “no S1 are P1”, the minor term is “some P1 are S2 while no S2 are S1”, and the conclusion term is “there are no S1 which are S2”. This example needs the extra condition: “no S2 are S1”. An example, instance 9, is

9. if no male is a female and some females are sopranos while no sopranos are male, then no male is a soprano.

Here the subjects and predications merge. One could say that there are only subjects or only predications. Note: in the real world one could argue if there is a male

who is a soprano, then there is a soprano who is male, a contradiction—but that is not constructive. So we write a LINCOS text instead. Note: we avoid the use of existence functions in the following because of the negations in the instance.

```

CONSTANTS male, female : Set.
CONSTANT is-male : male → Prop.

CONSTANT m : male
      a particular male, useful but not needed
CONSTANT is-female : female → Prop.
CONSTANT f : female
      a particular female
CONSTANT is-soprano : female → Prop
      some female is a soprano, in fact f is a soprano

HYPOTHESIS m-not-f : (x : Set) (is-male x) →
~(is-female x).
      no male is female

HYPOTHESIS not-f-not-s : (x : Set) ~(is-female x) →
~(is-soprano x)
      if x is not a female x is not a soprano, e.g. substituting m for x

HYPOTHESIS s-not-m : (x : Set) (is-soprano x) →
~(is-male x)
      no soprano is male

FACT f.7.9.1 : (x : Set) (is-male x) → ~(is-soprano x)
      no male is soprano. Verification

[H : (x : Set) (is-male x)] (m-not-f H) : (x : Set)
(is-male x) → ~(is-female x).
[H : (x : Set) (is-male x)] (m-not-f H) (not-f-not-s x)
: ~(is-soprano x).

f.7.9.1 = [H : (x : Set) (is-male x)] (m-not-f H)
(not-f-not-s x) : ~(is-soprano x).

```

Note. In chap. 2 we mentioned that for any $X : \text{Prop}$, $\sim X$ can be replaced with $(\text{not } X)$.

Also $(\text{not } X) = X \rightarrow \text{nil}$. So $(x : \text{Set})(\text{is-male } x) \wedge \sim(\text{is-female } x)$, can be replaced by $(x : \text{Set})(\text{is-male } x) \wedge (\text{is-female } x) \rightarrow \text{nil}$. We could have used this here.

The proof of the LINCOS fact f.7.9.1 is a verification of an instance of *Ferio*.

Reference

A. Ollongren *Aristotelian Syllogisms*, Acta Astronautica 68 (2011), 549–553

Chapter 8

Wittgenstein's Theatre

Intention

We have indicated in the previous chapters that verification of facts is effectuated essentially in a mechanical way—all of the processes, the easy ones but also the intricacies of the more difficult ones, can be carried out by some sort of (computing—data processing) machinery. This observation is important. Receivers of messages coded in LINCOS will be faced with a rather serious decoding and interpretation problem. Because machinery is involved in the construction of messages, there are interesting, promising perspectives for the decoding problem. We have remarked before that one can imagine that the receivers might be in the position to let the hard work be done by some sort of (intelligent) computing machinery or artefacts of for us unknown types. In that case the receivers might be able to program their machines in such a way that the LINCOS annotations of our messages can be treated (perhaps even interpreted) by them. In order to get some perspective on this aspect of the interstellar communication problem, we provide in this chapter a somewhat simplified outline of how machinery can do the job of verification.

Introduction

The following is the first part of Bertrand Russell's *Introduction* (1922) to Wittgenstein's *Tractatus Logico-Philosophicus* (1921).

- "Mr Wittgenstein's *Tractatus Logico-Philosophicus*, whether or not it prove to give the ultimate truth on the matters with which it deals, certainly deserves, by its breadth and scope and profundity, to be considered an important event in the philosophical world. Starting from the principles of Symbolism and the relations which are necessary between words and things in any language, it applies the result of this inquiry to various departments of traditional philosophy, showing in each case how traditional philosophy and traditional solutions arise out of ignorance of the principles of Symbolism and out of misuse of language.

The logical structure of propositions and the nature of logical inference are first dealt with. Thence we pass successively to Theory of Knowledge, Principles of Physics, Ethics, and finally to the Mystical (*das Mystische*).

In order to understand Mr Wittgenstein's book, it is necessary to realise what is the problem with which he is concerned. In the part of his theory which deals with Symbolism he is concerned with the conditions which would have to be fulfilled by a logically perfect language. There are various problems as regards language. First, there is the problem what actually occurs in our minds when we use language with the intention of meaning something by it; this problem belongs to psychology. Secondly, there is the problem as to what is the relation subsisting between thoughts, words, or sentences, and that which they refer to or mean; this problem belongs to epistemology. Thirdly, there is the problem of using sentences so as to convey truth rather than falsehood; this belongs to the special sciences dealing with the subject-matter of the sentences in question. Fourthly, there is the question: what relation must one fact (such as a sentence) have to another in order to be *capable* of being a symbol for that other? This last is a logical question, and is the one with which Mr Wittgenstein is concerned. He is concerned with the conditions for *accurate* Symbolism, i.e., for Symbolism in which a sentence "means" something quite definite. In practice, language is always more or less vague, so that what we assert is never quite precise. Thus, logic has two problems to deal with in regard to Symbolism: (1) the conditions for sense rather than nonsense in combinations of words; (2) the conditions for uniqueness of meaning or reference in symbols or combinations of symbols. A logically perfect language has rules of syntax which prevent nonsense, and has single symbols which always have a definite and unique meaning. Mr Wittgenstein is concerned with the conditions for a logically perfect language—not that any language is logically perfect, or that we believe ourselves capable, here and now, of constructing a logically perfect language, but that the whole function of language is to have meaning, and it only fulfills this function in proportion as it approaches to the ideal language which we postulate.

The essential business of language is to assert or deny facts. Given the syntax of language, the meaning of a sentence is determined as soon as the meaning of the component words is known. In order that a certain sentence should assert a certain fact there must, however the language may be constructed, be something in common between the structure of the sentence and the structure of the fact. This is perhaps the most fundamental thesis of Mr Wittgenstein's theory. That which has to be in common between the sentence and the fact cannot, he contends, be itself in turn *said* in language. It can, in his phraseology, only be *shown*, not said, for whatever we may say will still need to have the same structure.

The first requisite of an ideal language would be that there should be one name for every simple, and never the same name for two different simples. A name is a simple symbol in the sense that it has no parts which are themselves symbols. In a logically perfect language nothing that is not simple will have a simple symbol. The symbol for the whole will be a "complex", containing the symbols for the parts. In speaking of a "complex" we are, as will appear later, sinning against the rules of philosophical grammar, but this is unavoidable at the outset. "Most propositions and questions that have been written about philosophical matters are not false but senseless. We cannot, therefore, answer questions of this kind at all, but only state their senselessness. Most questions and propositions of the philosophers result from the fact that we do not understand the logic of our language. They are of the same kind as the question whether the Good is more or less identical than the Beautiful" (4.003). What is complex in the world is a fact. Facts which are not compounded of other facts are what Mr Wittgenstein calls *Sachverhalte*, whereas a fact which may consist of two or more facts is a *Tatsache*: thus, for example "Socrates is wise" is a *Sachverhalt*, as well as a *Tatsache*, whereas "Socrates is wise and Plato is his pupil" is a *Tatsache* but not a *Sachverhalt*." (end of quotation).

Verification Machinery

The present author cannot think of a better introduction to some of the fundamental problems facing the designer of a linguistic system for cosmic communication based on “Symbolism” (in Russell’s words, i.e. Symbolic Logic) with which the present treatise is concerned. We do not claim that the LINCOS system described in this book, i.e. the combination of sentences in natural language and annotations in the logical system chosen, is anything near a “perfect logical language” (using Russell’s words), not even for the purpose of limited scope for which it was developed. But we feel that the design presented in this book has significant advantages over other proposals. Given a message for ETI in some form, be it text in a human language, a musical score, a picture or some combination of these, a partial explanation of the logic contents is supplied in the form of annotations. An important advantage is, as we have seen, that annotations in the form of declarations, definitions and facts are guaranteed to be correct, provided that the environment is sound. Another is that correctness can be verified by automatic reasoning machines, i.e. computers supplied with the necessary reasoning programware.

Facts in LINCOS are in Wittgenstein’s words *Tatsache*, and they are essentially provably correct implications. Examples have been elaborated in earlier chapters furnishing grounds for this claim. However, missing in those chapters is a somewhat more instructive and clearer exposé of the mechanics behind the process leading to correctness or, in other words, the *dynamics* of the verifications. We shall endeavour now to provide some insight into this matter. We do so by providing abstract examples (i.e. not annotations of texts as one would do in LINCOS), first trivial ones and then proceeding to non-trivial cases. Most of this is in the spirit of several of Wittgenstein’s theses. Some of these, relevant in our discussions, are listed here. Note that the introductory theses are

- 2. What is the case, the fact, is the existence of atomic facts.
- 2.01 An atomic fact is a combination of objects (entities, things).

Wittgenstein continues with

- 2.0272 The configuration of objects produces states of affairs.
- 2.03 In a state of affairs objects fit into one another like the links of a chain.
- 2.031 In a state of affairs objects stand in a determinate relation to one another.
- 2.032 The determinate way in which objects are connected in a state of affairs is the structure of the state of affairs.
- 2.033 Form is the possibility of structure.
- 2.034 The structure of a fact consists of the structures of states of affairs.
- 2.04 The totality of existing states of affairs is the world.
- 2.05 The totality of existing states of affairs also determines which states of affairs do not exist.
- 2.06 The existence and non-existence of states of affairs is reality. (We call the existence of states of affairs a positive fact, and their non-existence a negative fact.)
- 2.061 States of affairs are independent of one another.
- 2.062 From the existence or non-existence of one state of affairs it is impossible to infer the existence or non-existence of another.
- 2.063 The sum-total of reality is the world.

It seems not immediately clear how to represent Wittgenstein's "state of affairs" in LINCOS. In order to keep the discussion transparent, we will represent this "state of affairs" produced by objects (i.e. entities or terms in LINCOS) by a *stage* (the *environment* introduced before). Thereby we do not violate explicitly any of Wittgenstein's assertions 2.0272–2.032. But we also do not elaborate his 2.03, 2.031 and 2.032 within the stage. In other words we use the implicit conventions of the Calculus of Constructions to take care of any relations between the entities in the environment. Since we are concerned with a kind of dynamical processes when facts are to be verified, the stage changes stepwise as the proof process proceeds. Facts and their structures are the concepts used throughout this book. The structure of a fact (Wittgenstein's 2.034), or rather the structuring of a fact, is dictated by the proof process employed. This is the mechanical aspect. A fact can, once it is verified, be entered in the stage in the normal way and used when required. The stage corresponds to Wittgenstein's world satisfying 2.06, except that non-existence is absent in our case. Assertion 2.061 is satisfied. The cases 2.05 and 2.062 are not relevant.

Below the mechanics of verifications is shown by means of examples.

Setting the Stage

Let part of the initial stage be

CONSTANTS $A, B, C : \text{Prop}.$

As shown before logical conjunction and disjunction can be entered into the stage by inductive definitions. We use here an unparametrized formulation slightly different from the one in Chap. 2. Here A and B are used as representatives of residents of Prop .

INDUCTIVE $\wedge_2 : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop} := \text{Conj} : A \rightarrow B \rightarrow A \wedge_2 B.$
 INDUCTIVE $\vee_2 : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop} := \text{Prim} : A \rightarrow A \vee_2 B \mid$
 Sec : $B \rightarrow A \vee_2 B.$

The index 2 is added here to the operators $\wedge$ and $\vee$ to show the arity of them (it is of course also evident from their types $\text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$). For $\wedge_2$ there is exactly one induction hypothesis, in fact $A \rightarrow B$, for $\vee_2$ there are two, A and B .

Observe that in order to prove $A \wedge_2 B$ residents of both A and B are required. For $a : A, b : B$, we have $(\text{Conj } a \ b) : A \wedge_2 B$. Further observe that for $a : A$, fulfilling the first induction hypothesis, $(\text{Prim } a) : A \vee_2 B$; this is already an informal proof of $A \vee_2 B$. For $b : B$, fulfilling the second induction hypothesis, $(\text{Sec } b) : A \vee_2 B$; this too is an informal proof of it. In order to formally verify $A \vee_2 B$ it is required that both induction hypotheses are satisfied. This is because A and B occur symmetrically in the definition of $\vee_2$.

In the following case of arity three

INDUCTIVE $\backslash /_3 : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop} :=$
 $\text{Prim} : A \rightarrow A \backslash /_3 B \backslash /_3 C \mid \text{Sec} : B \rightarrow A \backslash /_3 B \backslash /_3 C \mid \text{Tri} : C \rightarrow A \backslash /_3 B \backslash /_3 C.$

the specific conclusion $A \vee_3 B \vee_3 C$ is verified if $a : A, b : B, c : C$ exist, because then each of the induction hypotheses is taken into account. On the other hand one cannot construct a resident of A (or B or C) if only a resident of $A \vee_3 B \vee_3 C$ is available. In other words

$(A \backslash /_3 B \backslash /_3 C \rightarrow A)$ and $(A \backslash /_3 B \backslash /_3 C) \rightarrow (B \text{ and } A \backslash /_3 B \backslash /_3 C) \rightarrow C$

is not the case if residents of A (or B or C) are not available.

Simple Facts Verified

Consider the following simple case:

FACT f.8.1 : $A \rightarrow A$.
 verified by f.8.1 = $[H : A]H$.

The construction of the verification involves successive extensions to the environment. We show here the steps involved. The environment (i.e. the stage) is written to the left of the symbol “ $\parallel$ ” and the part that remains to be proved to the right of it. As soon as that part is proved we write $\parallel -$. Changes to the stage are indicated on the left-hand side of the listing: usually introductions of new local variables and elimination steps. Together these form the series of steps needed for the verification. The process ends with *Quod erat demonstrandum*, QED, when a resident of the conclusion has been constructed. In the case of FACT f.8.1 we have

<i>Actions</i>	<i>Stage</i>
	$A : \text{Prop} \parallel A \rightarrow A$
Intro	$H : A \parallel - A$
QED	$H : A.$

Here is a less elementary case:

FACT f.8.2 : $A \wedge B \rightarrow A$.
 f.8.2 = $[H : A \wedge B] (\text{ELIMH } [h1 : A; h2 : B] h1).$

<i>Actions</i>	<i>Stage</i>
	$A, B : \text{Prop} \parallel (A \wedge B) \rightarrow A$
Intro	$H : (A \wedge B) \parallel A$
ELIM H	$h1 : A; h2 : B \parallel - A$
QED	$h1 : A.$

Note. The type of $(\text{ELIM H } [h1:A; h2 : B]h1)$ is A because the type of $h1$ is A . In Chap. 2 we mentioned this special behaviour of the function ELIM .

Much less elementary is a case containing redundancy, not made use of due to the mechanical nature of the verification process.

$\text{FACT } f.8.3 : (A \wedge B) \wedge (A \wedge C) \rightarrow C.$
 $f.8.3 = [H : (A \wedge B) \wedge (A \wedge C)]$
 $(\text{ELIMH } [h1 : A \wedge B; h2 : A \wedge C])$
 $(\text{ELIMh2 } [h21 : A; h22 : C] h22)).$

<i>Actions</i>	<i>Stage</i>
	$A, B, C : \text{Prop.}$
Intro	$H : (A \wedge B) \wedge (A \wedge C) \parallel C$
ELIM H	$h1 : A \wedge B; h2 : A \wedge C \parallel C$
ELIM h2	$h21 : A; h22 : C \parallel -C$
QED	$h22 : C.$

Less Simple Facts Verified

Now we are ready for showing the dynamics of verifications of facts involving propagation.

$\text{FACTJanus} : ((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C).$
 $\text{Janus} = [H1 : (A \rightarrow B) \wedge (B \rightarrow C); H2 : A]$
 $(\text{ELIMH1} : [h1 : (A \rightarrow B); h2 : (B \rightarrow C)] (h2 (h1 H2))).$

<i>Actions</i>	<i>Stage</i>
	$A, B, C : \text{Prop.}$
Intros	$H1 : (A \rightarrow B) \wedge (B \rightarrow C); H2 : A \parallel C$
ELIM H1	$h1 : (A \rightarrow B); h2 : (B \rightarrow C) \parallel -C$
QED	$(h2 (h1 H2)) : C.$

The fact was given the name Janus in honour of the Dutch artist Janus Nuiten, who entered the formula in his oil canvas portrait of the author, created in January 2012. The portrait is to be displayed in Leiden University's historical "Gallery of Professors through the ages".

$\text{FACT } f.8.4 : A \rightarrow (B \wedge C) \rightarrow ((A \rightarrow B) \wedge (A \rightarrow C)).$
 $f.8.4 = [H1 : A; H2 : B \wedge C]$
 $(\text{ELIM } H2 [h21 : B; h22 : C])$
 $(\text{Conj } A \rightarrow BA \rightarrow C [* : A] h21 [* : A] h22)) .$

<i>Actions</i>	<i>Stage</i>
	A, B, C : Prop.
Intros	H1 : A; H2 : (B ∧ C) ∥ (A → B) ∧ (A → C)
ELIM H2	h21 : B; h22 : C ∥- (A → B) ∧ (A → C)
QED	(Conj A → B A → C [* : A]h21 [* : A]h22): (A → B) ∧ (A → C).

Note: Here $[* : A]h21 : A \rightarrow B$ and $[* : A]h22 : A \rightarrow C$ are used.

The verifications of expressions involving $\vee$ are as a rule more complicated because of the general requirement that each induction hypothesis must be taken into account as explained above. Let us look first at the following case, seen before in Chap. 2.

```
FACT f.8.5 : (A \ / B) → (B \ / A).
f.8.5 = [H : A \ / B]
        (ELIMH [h1 : A] (Sec B A h1) [h2 : B] (Prim B A h2)).
```

Notes: In this example both the second and third arguments of ELIM are used for verification to be of type $B \vee A$ (the conclusion). Under ELIM H we have the following:

- The type of $[h1 : A](\text{Sec } B \ A \ h1)$ is that of $(\text{Sec } B \ A \ h1)$, so it is $B \vee A$.
- The type of $[h2 : B](\text{Prim } B \ A \ h2)$ is that of $(\text{Prim } B \ A \ h2)$, so it is $B \vee A$.

<i>Actions</i>	<i>Stage</i>
	A, B, C : Prop.
Intro	H : A ∨ B ∥ B ∨ A
ELIM H	h1 : A ∥- B ∨ A
QED	(Sec B A h1) : B ∨ A
	h2 : B ∥- B ∨ A
QED	(Prim B A h2) : B ∨ A

Much more complicated reductions are needed for verifying the following associative law.

```
FACT f.8.6 : (A \ / B) \ / C → A \ / (B \ / C) .
f.8.6 = [H : (A \ / B) \ / C]
        (ELIM H [h1 : A \ / B]
          (ELIM h1
            [h11 : A] (Prim A (B \ / C) h11)
            [h12 : B] (Sec A (B \ / C)
                      (Prim A (B \ / C) h12 )) )
          [h2 : C] (Sec A B \ / C (Sec B C h2) )) .
```

Notes: From ELIM H followed by $[h1 : A \vee B](\text{ELIM } h1 \ \dots)$ we have within $(\text{ELIM } h1 \ \dots)$

```
[h11 : A] (Prim A (B \ / C) h11) : A \ / (B \ / C)
```

$[h12 : B] \text{ (Sec A (B \setminus C) (Prim A (B \setminus C) h12)) : A \setminus (B \setminus C)}$
 using $(\text{Prim A (B \setminus C) h12}) : B \setminus C$.

From ELIM H followed by $[h2 : C]$... we have

$(\text{Sec B Ch2}) : B \setminus C$
 $(\text{Sec A B \setminus C (Sec B Ch2)}) : A \setminus (B \setminus C)$.

The dynamics of these reductions clarify the reductions.

<i>Actions</i>	<i>Stage</i>
	A, B, C : Prop.
Intro	H : (A $\vee$ B) $\vee$ C $\parallel$ A $\vee$ (B $\vee$ C) to be proved
ELIM H	h1 : A $\vee$ B $\parallel$ A $\vee$ (B $\vee$ C)
ELIM h1	h11 : A $\parallel$ - A $\vee$ (B $\vee$ C)
QED	(Prim A (BVC) h11) : A $\vee$ (B $\vee$ C)
	h12 : B $\parallel$ - A $\vee$ (B $\vee$ C)
QED	(Sec A (BVC) (Prim A (BVC) h12)) : A $\vee$ (B $\vee$ C)
	h2 : C $\parallel$ - A $\vee$ (B $\vee$ C)
QED	(Sec A BVC (Sec B C h2)) : A $\vee$ (B $\vee$ C)

Each of the examples displayed above is characterised by *decidability*, i.e. the proof machinery achieves a final result (QED) in a finite number of discrete steps. One can evidently image the existence of “facts” not admitting the requirement that verifications are reached in a finite process. Such “facts” are then by definition outside the realm of LINCOS. As an example consider the somewhat cryptic sentence:

If the sea is quiet the storm must have calmed down. If the foregoing is the case the sea is quiet. The foregoing implies that the sea is quiet.

That looks strange but is quite harmless. Let us agree on the representations.

A represents “the sea is quiet”.

B represents “the storm must have calmed down”.

The sentence is then represented by $((A \rightarrow B) \rightarrow A) \rightarrow A$. In classic two-value logic (each term has a truth value “true” or “false”) this expression is a tautology (the truth value of the expression is “true” for all combinations of truth value assignments for A and B). We have here Peirce’s law, see also Chapter 24. Yet we cannot write LINCOS annotations to the sentence. Given a stage where A and B are known as constants, let us see what actions can be taken for verifying.

FACT ? : ((A $\rightarrow$ B) $\rightarrow$ A) $\rightarrow$ A.

<i>Action</i>	<i>Stage</i>
	A, B : Prop.
Intro	h1 : ((A $\rightarrow$ B) $\rightarrow$ A) $\parallel$ A
Intro	h2 : (A $\rightarrow$ B) $\parallel$ A

Now the process gets stuck: There is no action available to yield a resident of A. Technically speaking the reason is that h1 is not coupled to an inductive form. Therefore the machinery has no access to induction hypotheses to resolve the premise $((A \rightarrow B) \rightarrow A)$.

Generalisation, More Advanced Verifications

Consider the sentence

If the sea is quiet the storm must have calmed down. *And* it is the case the sea is quiet. The foregoing implies that the storm must have calmed down.

The sentence is represented by $((A \rightarrow B) \wedge A) \rightarrow B$, and the premise is $((A \rightarrow B) \wedge A)$, so we have here Modus Ponens. In this example the premise is an inductive form, and the proof machinery does not get stuck (of course).

FACT f.8.7 : $((A \rightarrow B) \wedge A) \rightarrow B$

Action	Stage
	A, B : Prop.
Intro	$h : ((A \rightarrow B) \wedge A) \parallel B$
ELIM h	$h1 : A \rightarrow B; h2 : A \parallel B$
QED	$(h1\ h2) : B$

f.8.7 = $[h : ((A \rightarrow B) \wedge A)](\text{ELIM } h\ [h1 : A \rightarrow B; h2 : A](h1\ h2))$.

A generalisation of the actions of the proof machinery is obtained in discussing parametrized functions, introducing either λ -bound or all-quantised variables. For Modus Ponens in general form we have

FACT MP-gen : $(x, y : \text{Prop}) (x \wedge (x \rightarrow y)) \rightarrow y$.
 MP-gen = $[x, y : \text{Prop}; H : (x \wedge (x \rightarrow y))]$
 $(\text{ELIMH } [h1 : x; h2 : x \rightarrow y] (h2\ h1))$.

Actions	Stage
	$x, y : \text{Prop}$.
Intro	$H : x \wedge (x \rightarrow y) \parallel y$, to be proved
ELIM H	$h1 : x; h2 : x \rightarrow y \parallel y$
	QED $(h2\ h1) : y$

Finally here is the example of Modus Tollens including general variables of type Prop.

FACTMT-gen : $(x, y : \text{Prop}) (\sim y \wedge (x \rightarrow y)) \rightarrow \sim x$.
 MT-gen = $[x, y : \text{Prop}; H : (\sim y \wedge (x \rightarrow y))]$
 $(\text{ELIMH } [h1 : \sim y; h2 : x \rightarrow y; h3 : x] (h1\ (h2\ h3)))$.

Actions	Stage
	$x, y : \text{Prop.}$
Intro	$H : \sim y \wedge (x \rightarrow y) \parallel \sim x$, to be proved
ELIM H	$h1 : \sim y; h2 : x \rightarrow y; h3 : x \parallel x \rightarrow (h1 (h2 h3))$
QED	$[h3 : x](h1 (h2 h3)) : x \rightarrow \text{nil} (= \sim x)$.

Under ELIM H, the local λ -bound variables $h1$ and $h2$ are given the types “inherited” from the induction hypotheses of $\wedge$; $h3$, however, is introduced *ad hoc* in view of the conclusion $\sim x$ to be proved.

Notes on Computer Implementation

In the previous section some significant important basics of the verification machinery are explained. The treatment is far from complete—we could have extended it with, e.g., nested function definitions intertwined with logical operators, but we feel that this would merely have resulted in rather extensive but not particularly interesting technical discussions. That would have led the discourse too far astray from the goals of this book. It is tempting to continue from here with discussing some aspects of possible computer implementations of the dynamics shown, and develop for instance in LISP-like programming some functions needed for that. This would be in the spirit of the author’s early work on LINCOS, using functional programming as a carrier. Efforts in this direction, undertaken at the time, were put in parking in favour of the approach shown in the present treatise. As a consequence the author refrains from elaborating here on the challenging problems in computer science facing designers of this kind of verification machinery—even though it is realised that “they”, receiving ET’s, most certainly will have to deal with issues of this kind.

Complete computer implementations of the Calculus of Constructions with Induction based on Type Theory have been carried out some time ago and recently as well—and are generally available. A well-known computer implementation of the generalised proof machinery is the Proof Assistant, developed in France and known by the name Coq. Several generations of Coq have been published since the mid nineties and are available from INRIA. Coq 8.1 is to date the stable version. The author’s PhD student at Leiden University, Gertjan Kamsteeg, developed a system implementing CCI about 10 years ago (new at the time) and used it for the formalization of Process Algebra with Data. All of these implementations include Inductive Types.

References in Chap. 1.

Part III

Annotations in LINCOS

Introduction

As we understand physical laws today, the vast distances and time spans involved exclude any form of real-time interstellar communication. Each highly developed intelligent species in the Galaxy can be supposed to possess its own “natural” language with its own peculiarities in syntax, semantics, and notation, each of them foreign to another species. Sequential notation (or the possibility of using it) is conceivably a universal concept. Even character-based scripture used on Earth is sequential. Most other linguistic concepts, however, will certainly have been influenced strongly by the development of the societies in which they evolved. This should be the case even if it could be shown that natural languages are based on a universal grammar in the Chomsky-ian sense. Thus members of an extraterrestrial intelligent society will either not be able at all to learn an alien language (not only because of time-delays) or only with great effort and certainly not as smoothly as for instance humans learn their own natural languages.

These arguments indicate that an interstellar message formulated in a natural language spoken on Earth will *a fortiori* not be interpretable for receiving species—even if such a message carries some sort of key to its interpretation. It seems an unavoidable conclusion that some kind of cosmic language, a *Lingua Cosmica* (LINCOS) will have to be used for the purpose of nontrivial interstellar communication between intelligent species by means of modulated electromagnetic radiation.

The LINCOS system for explaining message content described in this book aims at providing a useful basis for interstellar communication. The central idea already mentioned is that an abstract of the contents of a given message should be expressed in logic of a suitable modality. The message is thus enriched by logical descriptions, in fact annotations of message content, making the pair “message–abstract” easier to understand for alien cultures. The choice of logic for this purpose is motivated by the view that logic can be considered to be a reasonable and useful common ground for communication between galactic symbolic species. One cannot expect a species without the power of logical reasoning to be able to interpret an interstellar meaningful message.

In the present part some examples, mutually widely different in character, are reviewed in order to show some of the possibilities and difficulties of this way of reflecting on the interpretation issue.

Chapter 9

Logic Contents of Texts

Intention

The present book is concerned with the problem of explaining to intelligent beings or their information-processing artefacts located elsewhere in the universe the logic contents of interstellar messages, formulated in a human language or otherwise. If an existing natural language is used it is of course one they do not know and cannot understand immediately; this would be a reason for our messages to contain information by which recipients may get to know the syntax (and part of the semantics) of the language employed. In addition it is not unreasonable to assume that the recipients of such messages have no operational linguistic system resembling in some way those employed by our species. Even though all natural languages as used on Earth may be able to express “everything”, also abstract notions, they cannot be used *ohne weiteres* for interstellar communication because of the reasons given. On the other hand abstract notions contained in some way in an interstellar message stand a good chance to be understood if they are in the realm of logic. The present chapter discusses an example of a text, written in one of the languages on Earth, and shows in what way the logic contents of it can be dealt with in a formal manner.

Considerations

It is not unreasonable to suppose that all sufficiently developed intelligent species know abstract logic and are not unfamiliar with the particular mode used in this book (shortly *constructive logic* based on *type theory*). There may be a multitude of conventions in notation possible and perhaps some variation in the basics for this theory used by intelligent species outside of Earth, but problems presented by that might be solvable. One reason is that the mentioned logic uses a sparse set of rules and admits its own interpretation. Other aspects might be helpful too. For the formation of (typed) expressions it is *not* assumed that concrete objects like truth values (Booleans) and sets exist. All expressions are abstractly typed. There are only three

abstract basic types (constants free of interpretation, see Chaps. 1 and 2) together with the mapping concept available as basic instruments for constructing expressions. The set of primitives is rather small. On the other hand the expressive power of the logic, i.e. the ways and means of explaining textual content, is large.

There is, however, an intriguing question: Does the simplicity of the underlying logic make LINCOS easily applicable for given texts? To illustrate the ease (or difficulty) of doing so a summary of an ancient text from Plato's dialogues in *The Republic* is selected. We shall describe the logic content of it in the abstract setting mentioned. The present treatment is a major revision of an earlier paper by the author and D. Vakoch (Ollongren and Vakoch 2011).

An Ancient Text

In order to illustrate in a more concrete sense the idea of using a *meta* level for explaining the logic of message content, we use here a summary of a rather large ancient text in the Great Dialogues of Plato (427–347 B.C.). The first book from *The Republic* contains a dialogue between several persons on the topic of *justice*. Various definitions of the concept are proposed, but even Socrates (469–399 B.C.) fails to obtain a satisfactory form. The summary of book I, in the words of W.H.D. Rouse written in 1956, reads as follows:

Polemarchos invites Socrates and Glaucon to visit his father Cephalos' house. Various other friends are there as well. Cephalos talks about old age: eventually the conversation turns to the subject of justice. How do you define justice? asks Socrates. Polemarchos puts forward Simonides' definition—to render what is due [obligation]—but this on examination proves unsatisfactory. Here Thrasymachos breaks in, maintaining that the whole conversation so far has consisted of nothing but pious platitudes. Justice, he says, is whatever suits the strongest best. Might is right. A ruler is always just. Socrates suggests that even a ruler sometimes makes a mistake, and orders his subjects to do something which is really not to his advantage at all. Thrasymachos answers that in so far as he is mistaken he is not a true ruler. Socrates then argues that a doctor is primarily concerned to heal the sick, and only incidentally to make money: similarly, medicine seeks not its own advantage but the advantage of the human body. By analogy a ruler seeks the advantage of his subjects, not of himself. Thrasymachos then rushes off on a new tack. Injustice, he says, is virtuous, and justice is vicious. Justice is everywhere at the mercy of injustice, which is reviled not because men fear to do it but because they fear to suffer it. Socrates sets out to disprove this view, and establishes that justice is apparently wise and virtuous, and at the same time more profitable than injustice. But, he says, he is still without a definition of justice.

Message Content

All expressions in this chapter are written in the conventions explained earlier. The expressions are type expressions and are easily recognizable. Each expression “has a

type”—or “is of a type”—as mentioned before, see Chap. 1. For the statement that E is of type T, i.e. $E : T$, one can say that E “lives” in T, or that E is a “resident” of T. Alternatively one might state that E *exists* (see Chap. 2 and especially Chap. 23). We have seen that three distinct constants are predefined in a hierarchy:

Type—abstract types, with $Type : Type$, (remarkable!)
 Set—general entities, with $Set : Type$, also used in the present context
 Prop—logical propositions, with $Prop : Type$, here the most important type.

In Rouse’s text three distinct versions occur of what characterizes justice: the views of Simonides, Thrasymachos and Socrates. One might introduce three sections to keep the views apart (Ollongren and Vakoch 2011), but here we shall introduce three separate individuals and annotate their explanations. To begin with we shall need two inductively defined entities:

```
INDUCTIVE nil : Prop := .
INDUCTIVE unit : Prop := void : unit.
```

nil has no residents, unit has exactly one, and the selector (constructor) void.

(Note: void is introduced as a resident of nil in Chap. 3, for a different purpose.)

These two entities will be used in a binary coding for keeping the three persons Simonides, Thrasymachos and Socrates apart from each other.

```
INDUCTIVE person : Prop :=
  Simonides : (nil → unit) → person |
  Thrasymachos : (unit → nil) → person |
  Socrates : (unit → unit) → person.
```

Since we have for the types

```
Simonides : (nil → unit) → person.
Thrasymachos : (unit → nil) → person.
Socrates : (unit → unit) → person.
```

the three persons are neatly distinguished from one another. Note: In Chap. 14 we use a different way of distinguishing individuals from one another.

Using the following constants

```
CONSTANT is-Simonides : ((nil → unit) → person) →
Prop.
CONSTANT is-Thrasymachos : ((unit → nil) → person) →
Prop.
CONSTANT is-Socrates : ((unit → unit) → person) →
Prop.
```

we find

```
(is-Simonides Simonides) : Prop.
(is-Thrasymachos Thrasymachos) : Prop.
(is-Socrates Socrates) : Prop.
```

Note that the residents *a priori* supplied so far to exist in Prop are concerned with the occurring individuals only. In view of the text mentioning “unsatisfactory” and “best” we need two more entities—introduced as constants:

```
CONSTANT unsatisfactory : Prop.
CONSTANT best : Prop.
```

Simonides’ Definition of Justice

We need

```
CONSTANT obligation : Set.
HYPOTHESIS render : obligation → Prop.
```

and the following definition

```
DEFINE justice: (is-Simonides Simonides) → obligation
→ Prop :=
[y : (is-Simonides Simonides); z : obligation] (render
z).
```

render is introduced as a type which couples residents of obligation (the actual obligations) to residents of Prop (propositions). The coupling for an actual case is written (render x). And so

```
[z:obligation] (render z) : obligation → Prop.
```

is a way of stating that any (given) obligation z must be rendered. Formally y and z are introduced as local variables *via* lambda abstractions (see Appendix D). Note that the present treatment of actual obligations is related to formalization of the notion of altruism, see Chap. 12.

Above definition of justice is indeed unsatisfactory because there are no restrictions over obligations (examples in the first book from *The Republic*).

Thrasymachos’ Definition of Justice

The views of Thrasymachos are of a different kind. We need

```
CONSTANT suits-strongest-best : Set.
CONSTANT might : Set → Prop.
CONSTANT right : Set → Prop.
DEFINE might-is-right : Set → Prop :=
[x : Set] (might x) → (right x).
FACT suit : strongest → Prop → Prop.
```

This is an *introduction lemma*, which couples residents of strongest to the type $\text{Prop} \rightarrow \text{Prop}$, not just to Prop , for technical reasons. The proof of the lemma is omitted and we could have used a hypothesis. The definition of might-is-right implies

```
DEFINE (might-is-right suits-strongest-best) =
  (might suits-strongest-best)  $\rightarrow$  (right suits-
    strongest-best).
```

Note that the non-reflexive expression “might is right” is represented correctly. Finally the definition of justice is

```
DEFINE justice: (is-Thrasymachos Thrasymachos)  $\rightarrow$ 
  suits-strongest-best  $\rightarrow$  Prop :=
  [y : (is-Thrasymachos Thrasymachos);
   z : suits-strongest-best)](might z).
```

In this view there are no obligations; instead there is the qualification best (in the view of the strongest). Note that (right z) is implied because of the definition of might-is-right. The notion strongest is not explicit but incorporated in suits-strongest-best.

Thrasymachos' expressions “a ruler is always just” and “a mistaken ruler is not a true ruler” are not detailed yet. In view of Socrates' argument, a ruler can make mistakes, we use here an inductive definition for kind-of-Ruler (*cf.* inductive definitions in Chap. 2).

```
CONSTANT Ruler : Set.
CONSTANT is-just : Ruler  $\rightarrow$  Prop.
CONSTANT is-mistaken : Ruler  $\rightarrow$  Prop.
CONSTANT is-true : Ruler  $\rightarrow$  Prop.
INDUCTIVE kind-of-Ruler[R : Ruler] : Prop :=
  just : (is-just R) | mistaken : (is-mistaken R).
```

Here we use a shortened notation for the inductive type. This means for the two selectors

```
just : (R : Ruler) (is-just R).
mistaken : (R : Ruler) (is-mistaken R).
```

Note that because we use here the “either—or” conventions expressed by the token |, the two kinds of rulers are kept apart. Thrasymachos' judgements are

- a ruler is always just

```
DEFINE true-Ruler : (Ruler  $\rightarrow$  Prop) := [R : Ruler]
  (is-just R)  $\rightarrow$  (is-true R).
```

- a mistaken ruler is not a true ruler

```
DEFINE not-true-Ruler : (Ruler  $\rightarrow$  Prop) := [R : Ruler]
  (is-mistaken R)  $\rightarrow$   $\sim$ (is-true R).
```

Relations between these definitions could be elaborated but the necessary changes and improvements to be achieved are not interesting enough to justify the effort. So we leave the views of Thrasymachos.

Socrates' Definition of Justice

The most interesting views in the text are those of Socrates. It remains to be seen whether we need (is-Socrates Socrates) : Prop. Socrates argues in the text that a doctor is primarily concerned with healing the sick, and only incidentally to make money. So we introduce a doctor with an inductive definition, keeping the two cases separate using WITH (implying INDUCTIVE), as follows:

```
INDUCTIVE doctor : Prop :=
IDD : primarily → incidentally → doctor
    WITH primarily : Set := IDP : healing → sick →
        primarily
    WITH incidentally : Set := IDI : making-money
        → incidentally.
```

Above expresses Socrates' argument—in the form of an inductive definition of doctor. We use the particular recursive form shown here because the two characterizations of primarily and incidentally (both using recursion) are elegantly represented in this way. Recursion can be essential in inductive definitions—therefore it is discussed in this book separately (in Chap. 4). However, in the present discussion the recursive nature of a doctor and the three selectors IDD, IDP and IDI identifying three terms is not essential.

Separate definitions are needed for healing, sick and making-money. Each of these can be considered as abstract entities, for which no details need to be given, so they can be defined as members of Prop. They must be defined before they are used in doctor.

```
CONSTANTS healing, sick, making-money : Prop.
```

Next an introductory constant is formulated. Using it we define wise-doctor, the one who heals the sick.

```
CONSTANT doctor-heal-the-sick : healing → sick →
doctor → Prop.
DEFINE wise-doctor : healing → sick → doctor → Prop :=
    [h:healing; s:sick; d:doctor] (doctor-
        heal-the-sick h s d).
```

Socrates establishes that justice is both wise and virtuous. Therefore it is useful to introduce now virtuous-doctor, for the sake of simplicity the one who does not

make a lot of money. The entity money can be taken to be the same as the natural numbers, so

```
DEFINE money := nat,
```

where implied `nat : Set` will do. First another introductory constant

```
CONSTANT doctor-make-money : money → doctor → Prop.
```

Now virtuous-doctor can be defined:

```
DEFINE virtuous-doctor : money → doctor → Prop :=
  [m:money; d:doctor] ~ (doctor-make-
    money m d).
```

Note the important negation `~`. Of course wise and virtuous doctors might exist:

```
DEFINE wise-and-virtuous-doctor : healing → money →
  doctor → sick → Prop :=
  [h:healing; m:money; d:doctor; s:sick] ((wise-doctor h
  s d) /\ (virtuous-doctor m d)).
```

Now the stage is set for proving a theorem which says that a wise and virtuous doctor heals the sick and does not earn money (!):

```
FACT just-doctor : (h:healing; m:money; d:doctor;
  s:sick)
  (wise-and-virtuous-doctor h m d s) →
  (doctor-heal-the-sick h s d) →
  ~(doctor-make-money m d).
```

The lemma is self-explanatory, so the proof (not so elementary) is not given here. Should it be required to explain ingredients of this kind of proofs in message construction, many more extensive examples need to be supplied in the text. Above example shows, however, how to express the conditions for doctors to be wise and virtuous. Such doctors are seen to be just (the fact `just-doctor`).

Socrates then concludes that justice is apparently wise and virtuous, but indeed no definition has been obtained. The same applies to the annotations in logic, where only just is mentioned, embedded in the last mentioned fact.

Note that `(is-Socrates Socrates) : Prop` is not needed.

In much the same way as shown above, the views of Thrasymachos on second thoughts can be formalized in LINCOS, but that would not add to the insights provided by the above examples.

Discussion

In the foregoing typing exercise some selected words in an example text (a summary of one of the Great Dialogues of Plato) are literally quoted. That practice helps recipients of a message containing text and type expressions to recognize that logic is involved. At the same time the sequence of statements, as they are arranged in the text, has been kept more or less in the annotations supplied, but this may not always be feasible. Some but not all statements in the text are formalized. No effort has been made to treat separate sentences *an sich*. This is because it is not our aim to provide semantics for sentences (or other composite utterances) in natural language. Instead we wish to show how to clarify the logic contents of texts by (partially) supplying them with types.

Note that completeness is not aimed at. In our opinion only those aspects possibly helpful in clarifying logic textual contents (as judged by the “logician in charge” for interstellar message construction) should be taken into consideration for formalization. In this particular case the logical contents of the text are in fact rather meagre. The next chapter considers a case with much richer contents.

References

- A. Ollongren and D. A. Vakoch *Typing logic contents using Lingua Cosmica*, Acta Astronautica 68 (2011), 535–538
Great Dialogues of Plato translated by W.H.D. Rouse, Mentor Books (1956)

Chapter 10

An Astrolinguistic Experiment

Intention

One aim of studies on the topic of communication with extraterrestrials (CETI) is, from the point of view in this book, the creation of vehicles for the purpose of constructing ETI (who have their own “knowledge of the world”, most likely rather different from ours) interpretable messages of considerable size. Let the universe of discourse consist of texts from international literature, describing aspects of human society, using natural language. In view of the mentioned aims there is a *communis opinio* in the field of research on CETI that these should be supplemented with extra-linguistic information (abstract or concrete, e.g. pictograms represented by bitmaps). This is because it cannot be assumed that the descriptive sentences themselves are understandable immediately (*ipso tempore*) without great effort (*sine magno conatu*). This means that the introduction of a second (*meta*) level should be profitable, where some clarifying descriptive method can be used. By means of an experiment this aspect is scrutinized and evaluated in this chapter.

About the Second Level

The mentioned second level serves a dual purpose: partly clarified are on the one hand grammatical aspects of the language employed, and on the other hand more in detail the information content of messages. In this book we use constructive logic formulated in terms of computer implementable formalisms, also with in mind the possibility that ETI might use artefacts to do much of the tedious decoding tasks. This implies essentially as mentioned before that logical contents of messages (also considered to represent the deep structure of texts) should be described at some other level in a multilevel system as proposed here.

The question then presents itself whether a message for ETI thus constructed is indeed interpretable. The text itself being (presumably) impossible to understand, one might hope that “explanations” supplied at the second level will provide some kind of

“key” to understanding not only the message content but some grammatical aspects of the language employed as well. This hypothesis evidently cannot be tested in practice as long as we do not have “real-time” communication with ETI, or something resembling it. Long before that stage is reached (if ever) it is important to gain some insight into the possibilities and difficulties of interpretation using the multilevel approach advocated in this book. The present chapter describes an experiment designed to give an understanding of these aspects using a two-level approach. To the knowledge of the author such an experiment has not been carried out before.

The Experiment

In earlier chapters of this book the advantages of employing constructive logic at the second (*meta*) level are pointed out. The scope is necessarily limited to the logic contents of texts in some natural language placed in the first level, but there is the important advantage that formal logic expressions can be verified for correctness (if necessary also by computer programs) so that inherent errors in transmitted terms will not occur. In addition the verifications may serve as the meanings of these terms in Wittgenstein’s sense (see Chap. 8). It is reasonable to suppose that the receiver of transmissions annotated in LINCOS will be able to recognize the kind of logic employed, because of the simplicity of the associated formalism—which has at the same time large expressive power. By means of the meta-expressions supplied, ETI can be expected to not only gain some understanding of the text but in part also gain some insight in the grammatical structure of the language employed.

In order to (partially) test the latter expectations an experiment in CETI was carried out. One of the author’s graduate students at Leiden University’s Astronomical Department, Johanna Novozamsky, produced some time ago formal annotations to a rather substantial text of 74 sentences in the Czech language (Johanna Novozamsky 2000). Johanna N. is a native speaker of Czech (the author does not know that language) and supplemented the text with logical descriptions of the occurring objects and relations between them in terms of the French Coq system (see reference in Chap. 1). The present author, without knowing the “story” of the text, attempted to interpret the logic contents from the annotations. The present chapter discusses the possibilities and restrictions encountered during that task.

In order to restrict the size of this chapter, we reproduce and consider here only the first 18 of Johanna Novozamsky’s sentences, about a quarter of the original text, together with the annotations. Per sentence there are several logical clauses. These have been transposed into LINCOS forms. The clauses are mainly declarations (setting the stage so to speak) but two sentences give rise to lemmas (i.e. FACTs in LINCOS). For sake of clarity sentences from the Czech text are given first, while the logical descriptions at the second level are written using the format of LINCOS terms.

Note: The other way around, the transposition of a selected set of expressions from LINCOS to Coq can also be considered. This procedure would constitute a

means of achieving a partial, additional *validation* of our *Lingua Cosmica* system—in fact a tool to be used besides the method of self-interpretation and -verification and methods of assuring soundness of the environment (Ollongren 2012).

Here are Johanna Novozamsky's sentences and the clauses in logic she added as annotations.

1. Loutky, Role, Tvořičky, a Roky Prvního Vistoupení jsou Skupiny.

```
DEFINE Skupina := Set.
CONSTANTS Loutka, Role, Tvořiči : Skupina.
CONSTANT Rok-Prvního-Vistoupení : Skupina.
```

2. Spejbl, Hurvínek a Žeryk jsou Loutky.

```
CONSTANTS Spejbl, Hurvínek, Žeryk : Loutka.
```

3. Pes, Táta, a Syn jsou Role.

```
CONSTANTS Pes, Táta, Syn : Role.
```

4. Gustav Nosek a Josef Skupa, Gustav Nosek, a Karel Nosek a Josef Skupa jsou Tvořičky.

```
CONSTANTS Gustav-Nosek-a-Josef-Skupa, Gustav-Nosek :
Tvořiči.
CONSTANT Karel-Nosek-a-Josef-Skupa: Tvořiči.
```

5. 1920, 1930, a 1926 jsou Roky Prvního Vistoupení.

```
CONSTANTS 1920, 1930, 1926 : Rok-Prvního-Vistoupení.
```

6. loutka má v sobě Loutku, Roly, Tvořiče a Rok Prvního Vistoupení.

```
INDUCTIVE loutka : Set := ID : (Loutka → Loutka) →
(Role → Role) →
(Tvořiče → Tvořiče) →
(Rok-Prvního-Vistoupení → Rok-Prvního-
Vistoupení) → loutka.
```

7. Jméno Loutky je nějaká Loutka.

```
CONSTANT Jméno-Loutky : Loutka → Prop.
```

8. Jméno Role je nějaká Role.

```
CONSTANT Jméno-Role : Role → Prop.
```

9. Jméno Tvořiče je nějaký Tvořičky.

```
CONSTANT Jméno-Tvořiče : Tvořič → Prop.
```

10. Jméno Roku Prvního Vistoupení je nějaký Rok Prvního Vistoupení.

```
CONSTANT Roku-Prvního-Vistoupení : Roku-Prvního-
Vistoupení → Prop.
```

11. Kdyš jedna Loutka má nějakou Roly, tak ta Loutka nemá jinou Roly a jiná Loutka nemá tu jednu Roly.

HYPOTHESIS jedna-Loutka-jedna-Role :
 (x,y,z : Loutka) (a,b,c : Role)
 (Jméno-Loutky x) /\ (Jméno-Role a) →
 (~ (Jméno-Loutky x) /\ (Jméno-Role b)) /\
 ~ (Jméno-Loutky x) /\ (Jméno-Role c)) /\
 ((Jméno-Loutky y) /\ ~ (Jméno-Role a)) /\
 (Jméno-Loutky z) /\ ~ (Jméno-Role a) .

12. Loutka Hurvínek má Roly Syna.

FACT Hurvínek-je-Syn :
 (Jméno-Loutky Hurvínek) /\ (Jméno-Role Syn) →
 (~ (Jméno-Loutky Hurvínek) /\ (Jméno-Role Pes)) /\
 (~ (Jméno-Loutky Hurvínek) /\ (Jméno-Role Táta)) /\
 ((Jméno-Loutky Spejbl) /\ ~ (Jméno-Role Syn)) /\
 ((Jméno-Loutky Žerik) /\ ~ (Jméno-Role Syn)) .

13. Takže Loutka Hurvínek nemá Roly Psa.

HYPOTHESIS Hurvínek-není-Pes :
 ~ (Jméno-Loutky Hurvínek) /\ (Jméno-Role Pes) .

14. Takže Loutka Hurvínek nemá Roly Táty.

HYPOTHESIS Hurvínek-není-Táta :
 ~ (Jméno-Loutky Hurvínek) /\ (Jméno-Role Táta) .

15. Takže Loutka Spejbl nemá Roly Syna.

HYPOTHESIS Spejbl-není-Syn :
 (Jméno-Loutky Spejbl) /\ ~ (Jméno-Role Syn) .

16. Takže Loutka Žeryk nemá Roly Syna.

HYPOTHESIS Žeryk-není-Syn :
 (Jméno-Loutky Žeryk) /\ ~ (Jméno-Role Syn) .

17. Kdyš jeden Tvoříč udělal loutky která měla jeden Rok Prvního Vistoupení tak ten Tvoříč neudělal loutky která měla jinej Rok Prvního Vistoupení a loutka s jiným Rokem Prvního Vistoupení nebyla udělaná tím Tvoříčem.

HYPOTHESIS jeden-Tvoříč-jeden-Rok-Prvního-Vistoupení :
 (x, y, z : Tvoříč) (a,b,c : Rok-Prvního-Vistoupení)
 (Jméno-Tvoříče x) /\
 (Jméno-Roku-Prvního-Vistoupení a) →
 (~ (Jméno-Tvoříče x) /\
 (Jméno-Roku-Prvního-Vistoupení b)) /\
 (~ (Jméno-Tvoříče x) /\

```
(Jméno-Roku-Prvního-Vistoupení c)) /\
(Jméno-Tvořiče y) /\
~(Jméno-Roku-Prvního-Vistoupení a)) /\
((Jméno-Tvořiče z) /\
~(Jméno-Roku-Prvního-Vistoupení a)).
```

18. Tvořiče Gustav Nosek a Josef Skupa udělaly loutku která měla 1930 jako Rok Prvního Vistoupení.

```
FACT Gustav-Nosek-a-Josef- Skupa-dělaly-v-1930:
(Jméno-Tvořiče Gustav-Nosek-a-Josef Skupa) /\
(Jméno-Roku-Prvního-Vistoupení 1930) →
(~(Jméno-Tvořiče Gustav-Nosek-a-Josef Skupa) /\
(Jméno-Roku-Prvního-Vistoupení 1920)) /\
(~(Jméno-Tvořiče Gustav-Nosek-a-Josef Skupa) /\
(Jméno-Roku-Prvního-Vistoupení 1926)) /\
(~(Jméno-Tvořiče Karel-Nosek-a-Josef Skupa) /\
(Jméno-Roku-Prvního-Vistoupení 1930)) /\
(~(Jméno-Tvořiče Gustav-Nosek) /\
~(Jméno-Roku-Prvního-Vistoupení 1930)).
```

Interpretation

At the outset several factors simplified (partial) interpretation. The most important of these is undoubtedly the fact that the “sender” of the message and the annotations in Coq (Johanna Novozamsky) as well as the “receiver” (the present author) are familiar with the constructive logic involved, and basically use the same conventions.

Verifications of facts (not shown here) were included in the set of annotations in (Johanna Novozamsky 2000) but they were not considered during interpretation. This was because it appeared that they carried no useful extra information for the limited set of sentences considered. Note, however, that this is not generally the case.

In addition interpretation was simplified by the fact that the Czech language is configurational: it distinguishes subject and predicate in sentences. So it seems likely that “jsou” has to do with the verb *esse* (Latin) and the word “a” with *et* (L). “a” in sentence four must also signify conjunction, but why it is embedded in the constants (referred to as compounds above) in the annotations on the second level is unclear. The interchanging of postfixes (e.g. “a”, “u”, and “y” in the case of “Loutk”) must be grammatical—reasons for using them are not clear.

Objects which might represent *actors* occur in the text, because they appear to be atomic (and because their denominations are given with capital first letters). Hierarchy between the objects is not difficult to understand from the introductory parts of the text.

“Skupina” is apparently the universe of discourse. In the universe “live” so-called residents (using the term from type theory which can mean, for the example consid-

ered, something like “elements of” or “contained in”), i.e. the four categories: Loutka, Role, Tvořič, and Rok-Prvního-Vistoupení (sentence 1). Each of them is of type Skupina. Loutka contains what might be three actors Spejbl, Hurvínek, and Žeryk (sentence 2). Role contains what might also be three actors (sentence 3) and the same applies to Tvořič (sentence 4), although the members of the latter are compounds. Sentence 5 introduces presumably dates (using our knowledge of the world: the Gregorian calendar), not used in sentences 6 ... 12.

Sentence 6 has no immediate evident interpretation since the inductively defined loutka is not used before “well-formedness” is defined further on in the text.

Sentences 7 ... 10 assign properties to the mentioned four categories, by mapping them to Prop. The prefix Jméno is used for that purpose. Jméno-Loutky, for instance, can only be applied to a member of Loutka, the first of the four categories (sentence 1).

The hypothesis formulated alongside sentence 11 gives the meaning of the sentence. A coupling (with explicit restrictions) between three residents, *x*, *y*, and *z*, of Loutka prefixed with Jméno and the postscript ending changed, supposedly actors, and those of Role also prefixed with Jméno. Consequently it seems now more reasonable to view the residents of Role as *actions* rather than actors, and the coupling one-to-one. This interpretation is strengthened by the lemma (sentence 12) which couples the actor Hurvínek to the action Syn and not to Pes and not to Táta, and besides Spejbl not to Syn and Žeryk not to Syn.

The Experience Learns ...

A striking aspect illustrated by this experiment is the rather large size of the list of annotations, much larger in size than the text itself. That one runs into large-size messages (always the case, a symptomatic aspect) is due to compactness of natural language in general. This feature allows much to be said with few words, using implicit knowledge of the world in the background.

The translations of the sentences became available to the present author only after the above analysis was completed. The reader can judge for him/herself from the translations of the sentences given below how well the interpretation fits the meaning of the text (no backtracking was done). From the comparison one learns that any informative text meant for transmission to ETI must be composed with extreme care so as to ensure as little freedom of interpretation as possible. Moreover, the experience from the present experiment illustrates the well-known fact that a message of short size with many actors and actions easily may give rise to erroneous interpretation.

Further results learned from the experiment can be summarized in a few statements as follows. From the meta descriptions (the annotations in LINCOS) much understanding of the textual contents could be achieved—but the text was far from completely understood. For better understanding extra-linguistic information would have been useful. The logical relations furnished helped in understanding some elementary grammatical aspects of the Czech language—but that was not the primary aim of the project. Additionally the experience showed that the task of supplying (in a way constructing) an interpretation to a text written in an unknown language, but supplied with

annotations, will nevertheless always be a time-consuming effort. Frequent “indoor” backtracking and consequently reconsiderations will be necessary, and are responsible for that. But this should be no surprise in view of the experiences of J.F. Champollion (1790–1832) in his efforts on the interpretation of hieroglyphs of ancient Egyptian.

On the other hand the experience in the case described in this chapter was indeed quite satisfying, as the interpretation was far, but indeed not too far off the mark!

Appendix

1. Puppets, Roles, Craftsmen, and Years of First Performances are groups.
2. Spejbl, Hurvínek, and Žeryk are Puppets.
3. Dog, Father, and Son are Roles.
4. Karel Nosek and Josef Skupa, Gustav Nosek and Josef Skupa, and Gustav Nosek are Craftsmen.
5. 1920, 1930, and 1926 are Years of First Performances.
6. A puppet consists of a Puppet, a Role, a Craftsman, and a Year of First Performance.
7. The Name of a Puppet is one of the Puppets.
8. The Name of a Role is one of the Roles.
9. The Name of a Craftsman is one of the Craftsmen.
10. The Name of a Year of First Performance is one of the Years of First Performances.
11. If a Puppet has a Role then that Puppet does not have the other two Roles and the other two Puppets do not have that Role. (A hypothesis.)
12. Puppet Hurvínek has the Role of the Son. (A fact.)
13. So Puppet Hurvínek does not have the Role of the Father. (A hypothesis.)
14. So Puppet Hurvínek does not have the Role of the Dog. (A hypothesis.)
15. So Puppet Spejbl does not have the Role of the Son. (A hypothesis.)
16. So Puppet Žeryk does not have the Role of the Son. (A hypothesis.)
17. If a Craftsman has made a Puppet with a certain Year of First Performance
 - Then that Craftsman has not made the Puppets with the other two Years of First Performance,
 - And the other two Craftsmen have not made the Puppet with that Year of First Performance. (A hypothesis.)
18. The Craftsmen Gustav Nosek and Josef Skupa have made the Puppet with 1930 as the Year of First Performance. (A fact.)

References

- Johanna Novozamsky *Communication with Extra Terrestrial Intelligence*, Unpublished report
 Leiden University Astronomical Department (2000) 31 pp.
 A. Ollongren *On the validation of Lingua Cosmica*, manuscript in *statu nascendi* (2012)

Chapter 11

Aspects of Truth

Intention

The view on aspects of truth expressed in *Lingua Cosmica* as discussed in the present chapter was formulated for the first time in an unpublished paper presented by the author at the National Logic Symposium “Truth in Language”, held in the Dutch Royal Academy of Sciences in Amsterdam in 2005. The reason to include the essence of the paper here is that it contains a comparison with the setup of Professor Freudenthal’s book on LINCOS (see the part HISTORY in the present book). The point of view of that language is that of a *Lingua Characteristica et Systematica*. In the LINCOS system proposed in the present treatise, notably the same view is present in the background (see also the section Postscriptum, Appendix E).

In Freudenthal’s work basic mathematical concepts are introduced by examples, to begin with natural and rational numbers and operations over them—in the present monograph these examples do not have the same prominent position. In the present chapter, however, it is shown how examples of this kind can be elegantly incorporated in the new *Lingua Cosmica*. Freudenthal’s book includes discussions of aspects of set theory and formal propositional and predicate calculus in logic, whereas the present book uses only constructive (or intuitionistic) formal logic as the main base. On the other hand sets and the two calculi can be embedded if necessary.

Teaching Truth

Freudenthal’s LINCOS is according to the designer, a language for “cosmic intercourse”, and it satisfies an important criterion: it can be taught and learned. It fits in fact in the traditional quests for a perfect universal language. See also the citations from a review by Bruno Bassi in the paragraph History in Appendix C of the present book.

How is the concept of *truth* handled in Freudenthal's monograph? The constants *true* en *false* are introduced via examples with natural and rational numbers, for example,

$2 + 3 = 3 + 2 = 5$, <i>true</i>	$2 \times 3 = 3 \times 2 = 6$, <i>true</i>
$2/3 = 3/2$, <i>false</i>	equality is supposed to be understood from context

Correct and wrong (*good* and *bad*, Ben and Mal) are introduced by examples in textual form of conversations between persons (*Homo Sapiens*). Let Ha en Hb represent two humans conversing on topics from *arithmetica* with one another. An extraterrestrial alien receiver of a message in LINCOS containing (parts of) the text of a conversation involving Ha and Hb may conceive an arbitrary interpretation of the latter objects (e.g. they might be artefacts, robots, intelligent beings, or something else). It is important, however, that Ha en Hb are recognized to be two distinct entities and that there is some kind of exchange of information between them (perhaps an exchange of thoughts, views, opinions, ... based on certain processes, not elaborated in any detail in the message). An example of a conversation is (on the left-hand side a shorthand notation in preparation for writing LINCOS forms):

Ha Inq Hb ?x x = $3/4 + 7/4$	Ha informs Hb what is x if x = $3/4 + 7/4$?
Hb Inq Ha !x x = $1/4$	Hb informs Ha x is $1/4$
Ha Inq Hb !x Mal	Ha informs Hb x is wrong (<i>bad</i>)
Hb Inq Ha !x x = $5/2$	Hb informs Ha x is $5/2$
Ha Inq Hb !x Ben	Ha informs Hb x is correct (<i>good</i>)

Freudenthal uses a special notation for the lines on the right-hand side. Here *good* and *bad* are not identical to *true* and *false*. In the next example, for instance,

Ha Inq Hb ?x x = $3/4 + 7/4$	Ha informs Hb what is x if x = $3/4 + 7/4$?
Hb Inq Ha !x x = $20/8$	Hb informs Ha x is $20/8$
Ha Inq Hb !x Mal	Ha informs Hb x is wrong (<i>bad</i>)

Mal "is said" because the reduction of $20/8$ to $5/2$ is not carried out. However, the following would be in accordance with the linguistic tradition set by Ludwig Wittgenstein in the *Tractatus Logico Philosophicus* (1921) en in Bertrand Russell en A.N. Whiteheads *Principia Mathematica* (1910)

Ha Inq Hb ?x x = $3/4 + 7/4$	Ha informs Hb what is x if x = $3/4 + 7/4$?
Hb Inq Ha !x x = $1/4$	Hb informs Ha x is $1/4$
Ha Inq Hb !x Mal	Ha informs Hb x is wrong (<i>false</i>)
Hb Inq Ha !x x = $20/8$	Hb informs Ha x is $20/8$
Ha Inq Hb !x Ben	Ha informs Hb x is correct (<i>true</i>)

As explained before the present author's LINCOS is a linguistic system based on logic and in the simplest case two levels are distinguished containing

- Some text formulated in some (possibly natural) language—in the present chapter using a shorthand notation, including mathematical formalism,
- Annotations on the logical contents of the text in constructive logic based on the type theory, whereas the notion of truth is replaced by a verification requirement.

Verification of Truth

Here is above example in the form of a small text, mainly in shorthand natural language and now using prefix notation for the mathematics.

Ha and Hb exchanging thoughts.
 x varies over the rational numbers.
 ?x, what is x, if $(= x (+ (/3\ 4) (/7\ 4)))$.
 !x, the result is $(= x (/5\ 2))$.

In the so-called *prefix* notation an operator (arithmetical or otherwise) is placed leftmost and is followed by 0, 1, or more arguments of it, corresponding to its arity. The symbol Inq, with different interpretations depending on the context, is replaced by “what is x” or “the result is” in the text.

A logical annotation to this text in our LINCOS would be embedded in an *environment*, containing in any case lists of occurring constants and variables with their *types*. An environment contains *type definitions*. In general an environment consists of declarations, hypotheses, and definitions of functions (maps, mathematically speaking). A map supplied with arguments is represented in prefix notation, for example $(= x\ y)$, in view of the example above. However, strictly speaking in pure constructive logic there is no map $=$. So we dutifully introduce it here for “local use”. The necessary constants and variables used in above text are entered into the environment by the following declarations.

```
CONSTANTS Ha, Hb : Set.
VARIABLE x : Rat.
CONSTANT + : Rat → Rat → Rat.
CONSTANT / : Rat → Rat → Rat.
CONSTANTS ?, !:Rat → Prop.
CONSTANT = : Rat → Rat → Prop.
```

VARIABLE is used here, but in this case there is no difference in interpretation between VARIABLE and CONSTANT, because the declared constants can be (and usually are) treated as variables. The declarators VARIABLE and CONSTANT are delimiters as well, of which there are many, easily recognizable. Set is a type, and *could* be interpreted as an abstraction of a mathematical set. Ha and Hb are of type Set (they do have the same type, but they are distinct from one another). x a rational number is of type Rat, itself assumed to be typed; +, /, and = are maps each expecting two arguments of type rational number (arity 2).

Examples of types involving Rat and Prop are

$(+ (/ 1 2)) : \text{Rat} \rightarrow \text{Rat}$	$(= 2) : \text{Rat} \rightarrow \text{Prop}$
$(+ (/ 1 2) (/ 1 3)) : \text{Rat}$	$(= 2 2) : \text{Prop}, (= 2 3) : \text{Prop}$
$(+ (/ 1 2) (/ 1 3) (/ 1 4))$ is undefined	$(= 2 3 4)$ is undefined

Prop is a type and *could* be interpreted as an abstraction of a proposition in ordinary propositional logic, in this case over arithmetic operations. Note that $(= 2 2)$ and $(= 2 3)$ have the same type, but *true* and *false* are not on the scene. Note further that $?$ and $!$ have the same type, and both are maps of arity 1. Also $(? 2)$ and $(! 2)$ have the same type, $(? 2) : \text{Prop}, (! 2) : \text{Prop}$.

We have seen before in various contexts : if A is a type and if $a : A$, so a is of type A , we say “ A is the case”. This corresponds to Wittgenstein’s “Was der Fall ist, die Tatsache”. One also uses the expression: “ A is *justified*”, or “ A is *justified* by a ”. Sometimes we say a exists (Chap. 23). In the case of a verification, if a has been constructed such that $a : A$, one says “ A has been *verified* (by a)”. This corresponds to the verification concept from intuitionistic logic.

In view of Freudenthal’s example we need a map with the name Inq.

```
DEFINE Inq : Set → Set → Prop → Rat → Rat → Prop :=
[h1, h2 : Set; p : Prop; x, y : Rat] (= x y).
```

Because of the type of the map we have of course

```
[h1, h2 : Set; p : Prop; x, y : Rat] (= x y) : Set →
Set → Prop → Rat → Rat → Prop.
```

The arity of Inq is 5, so Inq expects at most 5 arguments of the types indicated. The 5-tuple must be enumerated in the so-called lambda form as we have seen before

```
[h1, h2 : Set; p : Prop; x, y : Rat],
```

binding $h1$, $h2$, p , x , and y to the types shown. These variables can (but do not have to) appear in the so-called *body* of the map Inq, $(= x y)$ in above case.

When a map is supplied with arguments, we say that the map is *applied*. Reminder: an application yields a type using the body of the map with appropriate replacements of the variables with the arguments. Here follow some examples of applications of Inq.

Let r be a rational number that can be reduced to r' (the nominator and denominator have no factors in common). We write

```
(Inq Ha Hb ?x x r) : (= x r').
(Inq Ha Hb !x x r) : (= x r').
```

The following types are then valid (using the “normal” notation for rational numbers):

```
Inq1 = (Inq Ha Hb ?x x (+ 3/4 7/4) ) : (= x 5/2).
Inq2 = (Inq Hb Ha !x x 1/4) : (= x 1/4) .
Inq3 = (Inq Hb Ha !x x 10/4) : (= x 5/2).
Inq4 = (Inq Hb Ha !x x 20/8) : (= x 5/2).
```

Some facts in the form of implications are then verifiable.

FACT f.11.1 : Inq1 $\rightarrow$ (= x 5/2) .

Verification f.11.1 = [* : Inq1] (Inq Hb Ha !x x 5/2) .
because [* : Inq1] (Inq Hb Ha !x x 5/2) : Inq1 $\rightarrow$ (= x 5/2) .

Note that the symbol * (“does not matter”) gets type Inq1, but it is not required in Inq1 $\rightarrow$ (= x 5/2) , the type of f.11.1.

FACT f.11.2 : Inq2 $\rightarrow$ (= x 5/2) . Not verifiable.

FACT f.11.3 : Inq3 $\rightarrow$ (= x 5/2) .

Verification f.11.3 = [* : Inq3] (Inq Hb Ha !x x 5/2) .
because [* : Inq3] (Inq Hb Ha !x x 5/2) : Inq3 $\rightarrow$ (= x 5/2) .

FACT f.11.4 : Inq4 $\rightarrow$ (= x 5/2) .

Verification f.11.4 = [* : Inq4] (Inq Hb Ha !x x 5/2) .
because [* : Inq4] (Inq Hb Ha !x x 5/2) : Inq4 $\rightarrow$ (= x 5/2) .

The symbol $\rightarrow$ is again overloaded, and it has two interpretations. It is used in lemma’s (facts) indicating implication, and in definitions of functional mapping (maps), as we have seen before.

The symbol *true* is absent as its use is replaced by the concept of *verification*. According to intuitionistic (constructive) logic, an implication is verified if there is a method by which the conclusion (the part to the right of the symbol $\rightarrow$) of an implication can be constructed. In other words, the part to the right of the symbol $\rightarrow$ can be shown to be the case by constructive means. The notion *false* is absent unless explicitly introduced in some way. If constructions fail to verify the conclusion of an implication, it cannot be assumed to be incorrect. This is *Tertium non Datur*.

Enriching the Environment

Verified facts can be added to the environment and are then available for reasoning, building definitions, and constructing proofs of facts. If an expected result cannot be verified, it cannot be considered to be a fact and therefore cannot be added to the environment. Returning to the above examples

Inq1, Inq3, and Inq4 have the same type (= x 5/2) so
(eq-p (= x 5/2) Inq1 Inq3) : Prop
(eq-p (= x 5/2) Inq3 Inq4) : Prop

where the map eq-p is defined inductively by

INDUCTIVE eq-p[X : Prop; x, y : X] : Prop :=
eq-p-intro : (eq-p X x y) .

The entity `eq-p-intro`, usually called a *selector*, can also be considered to be a *constructor* because it is operationally a constructive element in the inductive definition; its type is given by

```
eq-p-intro: (X : Prop; x, y : X) (eq-p X x y) .
```

Note that $(X : \text{Prop}; x, y : X)$ is not a lambda form, and it is an abbreviation of
`forall X : Prop, forall x : X and forall y : X.`

Facts of a more general nature can be verified as well. An example is the next transitive relation.

```
FACT f.11.5 :  
(eq-p (= x 5/2) Inq1 Inq3) →  
(eq-p (= x 5/2) Inq3 Inq4) →  
(eq-p (= x 5/2) Inq1 Inq4) .
```

We omit the verification of this fact because methods used for facts of a more general nature like this one are technically involved.

Static Relations

In Chap. 4 we have discussed the case of the well-known Russian Matryoshka's considered to be unrestricted recursive structures, *dynamic* in a way. In the Amsterdam paper a specific (simple) kind of the Russian dolls was used in order to illustrate a way of handling *static* relations between objects. The relations discussed here are of a different kind than those of books lying on shelves treated in Chap. 2. The objects are now dolls:

Large dolls ML, middle-sized dolls MM, and small dolls MS.

If a is a ML-doll and b is a MM-doll, then b “is contained in” a .

If b is a MM-doll and c is a MS-doll, then c “is contained in” b .

On the basis of these relations we have: if a is a ML-doll and c is a MS-doll, then c “is contained in” a . Note that “to contain in” is a transitive relation.

```
CONSTANTS ML, MM, MS : Set.  
CONSTANT a : ML.  
CONSTANT b : MM.  
CONSTANT c : MS.  
CONSTANT is-ML : ML → Prop.  
CONSTANT is-MM : MM → Prop.  
CONSTANT is-MS : MS → Prop.  
DEFINE b'in'a : MM → ML → Prop := [x : MM ; y : ML]  
(is-MM x) /\ (is-ML y) .
```

We have then $(b'in'a\ b\ a) : \text{Prop}$, but $(b'in'a\ a\ b)$ is undefined.

Because $\wedge$ commutes, we can write for the body of the function

```
(is-ML y) /\ (is-MM x).

DEFINE c'in'b : MS → MM → Prop := [x : MS; y : MM]
(is-MS x) /\ (is-MM y).
DEFINE c'in'a : MS → ML → Prop := [x : MS; y : ML]
(- x) /\ (is-ML y).
```

This is the *environment* for the annotations in LINCOS to the given text. In the environment some facts are true.

```
FACT 3.3.6 : (b'in'a b a) → (is-MM b) /\ (is-ML a).
verifiable
FACT 3.3.7 : (isMM b) /\ (is-ML a) → (b'in'a b a).
verifiable
FACT 3.3.8 : (b'in'a b a) /\ (c'in'b c b) → (c'in'a c
a). verifiable
```

Note that there are no definitions of $a'in'b$, $b'in'c$, and $a'in'c$. Seemingly true but *not* verifiable are therefore

```
not a'in'b
not b'in'c
not a'in'c.
```

For a more extensive, general discussion of the Russian Matryoshka's, seen as recursive structures, see Chap. 4.

Super- and Subvenience

In the present treatise the concept of supervenience is discussed in some detail in the Postscriptum, Appendix E. There we find that ... “The notion formalizes the intuitive idea that one set of facts can fully determine another set of facts” and more in particular that “B-properties supervene on A-properties if for any two possible situations identical with respect to their A-properties, this implies that they are identical too with respect to their B-properties”. We find there also the (formally verified) conclusion that LINCOS descriptions (B-properties) supervene on reality (A-properties) or that reality subvenes on LINCOS.

That observation implies important obligations for designers of messages for ETI in LINCOS. Multi-interpretable messages are not only undesirable—they need to be uni-interpretable and moreover as much as possible correct representations of reality as we experience it. The LINCOS annotations are per definition uni-interpretable, but the requirement they should fulfil is that indeed subvenience of reality on them is achieved. This conclusion is of course fully in the spirit of the teaching of truth, and indeed in the set up of the linguistic system described in this book.

An entirely different aspect is the question what parts of reality would be reasonable to incorporate in messages for ETI. Some possibilities are discussed in some detail in Chaps. 10, 12, and 15. Chapter 23, however, indicates that inherent complexities of reality may result in less perspicuous sequences of annotations by means of formal LINCOS terms.

Chapter 12

Human Altruism

Intention

The present chapter, a contribution to discussions on the possible contents of messages for interstellar communication, is concerned with the problem of characterizing in logical terms some essential moralistic aspects of human altruism. The rationale for this is that, once a suitable characterization is achieved and agreed upon, the *Lingua Cosmica* formulation proposed in this book can be utilized in actual message construction involving altruism. In the present chapter we have collected the LINCOS terms needed—this time in notes at the end of the discourse, enabling “easy reading” of the descriptive parts. We strive to achieve a measure of self-containment in this chapter and therefore collect relevant information on the basics of LINCOS as well.

Introduction

We shall be concerned with the problem of describing in an astrolinguistic sense certain altruistic aspects of human behaviour. The rationale for taking up this challenge has been described in an article on communicating basic principles of reciprocal altruism by D. Vakoch and M. Matessa (Douglas 2011). They note that certain aspects of this kind of behaviour can be modelled in mathematical (game theoretical) terms and astrophysical terms (using the case of stellar evolution). In the latter example no game theory is involved, but there is deterministic loss of stellar mass—and by the way, enrichment of the interstellar material.

In the present chapter we adopt a different point of view and develop descriptions of aspects of human altruism in a logical sense. For such descriptions the multilevel approach could be used: natural language supplemented with annotations in logic (using *Lingua Cosmica*). But in the present chapter we do not particularly use the multilevel approach, and use mainly logic from the start.

A strong motivation for looking at the problem in this way is the already mentioned assumption (if not certainty) that a party at the receiving end of an interstellar communication channel cannot be supposed to be acquainted in any way with the structure of natural languages as they are spoken, written, and understood by earthlings. Interpretation of what we wish to explain is especially difficult if aspects of human behaviour, e.g. altruism, are conveyed using a language. Interstellar textual messages emitted by us from Earth perhaps may at the best help receivers to appreciate one or more of our languages and perhaps learn some aspects of them. But interpretation by aliens, even partially, of text concerned with abstractions as altruism written in natural language is an entirely different matter. We have argued before that in order to help intelligent extraterrestrials with the task of understanding the semantic contents of messages (not especially the subtleties of the grammars employed) texts used in interstellar communication should in general be supplemented with extralinguistic information. In the view of the present author this aspect is one of the cornerstones of astrolinguistics. This kind of information should preferably be formulated in an abstract logic framework, e.g. in the proposed *Lingua Cosmica*, based on the Calculus of Constructions with Induction (CCI), as explained in detail not only in earlier chapters but also elsewhere (Ollongren 2003).

In the present chapter we consider only two aspects of altruism, *duty* and *obligation*, formally to be modelled in the *lingua*. Aspects of empathy are closely related to altruism, but these are not discussed in the present context. Useful for the understanding of this chapter on its own merits is the following short resume of the basic concepts of the logic formalism used in LINCOS as follows:

All terms (expressions) are typed—so we use here strong typing. Global constants, represented by identifiers (names), themselves terms, are introduced by declaring their types. A term may have residents, i.e. entities typed by that term. If a term has a resident, it is justified. Locally bound variables are introduced as (lambda) abstractions giving their types, which specify their domains. An abstraction also specifies the lexicographic range of the variables. Mappings from domains to ranges are powerful expressive terms. These not only are considered to be *bona fide* functions but can also denote implications—depending on the context in which they appear. LINCOS contains mechanisms for defining functions (as in the calculus of constructions). Inductive definitions are admitted. *True* and *false* are absent, but an implication can be verified; it is proved if and only if a justification for it, i.e. a resident of it, can be constructed—many examples of such verifications are shown throughout this book.

Instruments used in verifications of facts, a *sich* elementary but fundamental, are constructive as in CCI. All expressions (i.e. introductions or declarations, definitions, and verifiable facts, in general logic terms) in the present chapter are written in the notation of the LINCOS system.

Altruism

The *Concise Oxford Dictionary* of the English Language describes altruism in this way: “Regard for others as a principle of action”. The *Webster Dictionary* mentions:

... unselfish concern for the welfare of others: opposed to egoism.

Because “others” are involved this means that the general notion of altruism considered as a logic relation over some structure of terms (understood to contain representations of the logic concept of humans) is not reflexive, i.e. self-altruism is excluded, and should be considered to be in the realm of egoism. It is implicitly assumed that altruism is not subjected to the strong notions of symmetry and transitivity. Further it is understood that actors and behaviour are involved.

Moralism

We consider here altruism as a kind of *moral behaviour* and carry it into the realm of philosophy. Altruism as well as egoism are thereby viewed as moral behaviour of persons, the actors—who carry out actions in relation to themselves (*ego*’s) and others (*alter*’s). The acts are concerned with ethical goods for the benefit of persons and others. Moral behaviour is realized in this view by the acts of a person under the assumption that (other) persons are beneficiaries. This rather vague concept outlined so far needs an amount of structure so that it can be handled in a logic sense. In order to achieve that, several decisions have to be made. As there are various options available the following is not the only way of modelling moralism.

Individuals performing acts of moral behaviour need not be introduced separately as types. Persons and others are therefore collectively introduced as introductory constants. Any act of moral behaviour is associated with ethical goods and can result (or *should* result) in a benefit for some person. (Ethical) goods will be supposed to *reside* in a set G (using a LINCOS term). If g is a goods we say that it is of type G , writing $g : G$, expressing that G is the case, because g is resident or a type justification of G . Benefits are supposed to reside in a set B , and we write $b : B$ for a benefit, giving at the same time B , considered to be a type, an interpretation. We note here once again that types can have many residents, i.e. many interpretations (justifications). The type B of benefits is supposed to possess three special residents designated by nil, moderate, and large (no benefit, some benefit, and much benefit). So we need some global introductions (or declarations). Together with some reminders the necessary declarations are shown in the Notes see ^{1,2}.

Given a goods and a benefit, an act of moral behaviour will be represented by coupling the goods and the benefit in this order to form the type $G \rightarrow B \rightarrow \text{Prop}$. The symbol $\rightarrow$ means here that a mathematical injective mapping is used. Prop, the type

of propositions, is added here for pragmatic reasons, see ³ in the Notes. Benefit for others is given the same type. So we have new constants, Act-by-person and Benefit-by-other, see ⁴ in the Notes. Benefit for self is assumed to be independent of ethical goods. If $g : G$ and

$b : B$, then Act-by-person can be *applied* to g ; an application to g can be followed by an application to b . We write for these

$$\begin{aligned} (\text{Act-by-person } g) & : B \rightarrow \text{Prop} \\ (\text{Act-by-person } g \ b) & : \text{Prop}. \end{aligned}$$

Because the type of $(\text{Act-by-person } g \ b)$ is seen to be Prop , this application interprets Prop . The interpretation is by no means unique: Prop can have many mutually distinct interpretations, as one would intuitively expect, and as explained before. The duality between a type and its interpretation is important as we have seen. Note that there is no need to express whether or not the activity represented by $(\text{Act-by-person } g \ b)$ is actually performed. We consider in this chapter only static relations.

Types of Moral Behaviour

Now a position is reached where types of egoistic and altruistic moral behaviour can be defined, see ⁵ in the Notes. We have chosen here an *inductive* definition introducing

Moral-behaviour, of type $G \rightarrow B \rightarrow \text{Prop}$, in such a way that four different kinds are distinguishable. The entities EGO-excl, EGO-large, EGO-moderate, and ALT-excl figure as *selectors* (also called *constructors*) in the definition. The first three are connected to egoistic behaviour and the last one is the most interesting one for the purpose of the present chapter. For all cases of moral behaviour it is assured that an act as well as a benefit is involved. The types of the four selectors are mutually different. For purely altruistic behaviour the benefit for self is empty as we have (see ⁶ in the Notes):

$$\begin{aligned} \text{ALT-excl} & : (\text{ALL } g : G; \ b : B) \ (\text{Act-by-person } g \ b) \rightarrow \\ & (\text{Ben-by-self nil}) \rightarrow \\ & (\text{Ben-by-other } g \ b) \rightarrow (\text{Moral-behaviour } g \ b). \end{aligned}$$

We will describe now how it can be decided that

- if for some g and b each of $(\text{Act-by-person } g \ b)$, (Ben-by-self nil) , and $(\text{Ben-by-other } g \ b)$ is the case
- then $(\text{Moral-behaviour } g \ b)$ is the case.

This is how it is done, in the form of a proof. Assume

$$\begin{aligned} h1 & : (\text{Act-by-person } g \ b); \ h2 : (\text{Ben-by-self nil}); \\ h3 & : (\text{Ben-by-other } g \ b). \end{aligned}$$

and conclude

$$(\text{ALT-excl } g \ b \ h1 \ h2 \ h3) : (\text{Moral-behaviour } g \ b).$$

Duty and Obligation

The above sections describe moral behaviour itself without reference to embeddings in situations in society. In the present section we show how relations with duty and obligation can be entered in the formal discussions.

As starting point we take the observation that persons in society have the duty to act in accordance with moral behaviour. That is registered in the form of an inductive hypothesis, see ⁷ in the Notes. Only three cases are distinguished involving

(Ben-by-self large), (Ben-by-self moderate), and (Ben-by-self nil). This is because these cases carry some aspect of altruism, simply because the benefit for the others is not nil whereas an exclusively egoistic action is not altruistic in any respect.

A duty becomes an obligation if there is a *means* for carrying out the duty. In the context of the present discussion, we must therefore assume the existence of two constants, goods (an ethical goods) and ben (some “real” benefit). The introductory constants are already in the Notes, see ² there. In the abstract setting we have here, it can be left unspecified what the ethical goods (type G) and what the benefits (type B) actually are for some given situation in society. This is one of the advantages of the point of view taken here.

The two introductory constants are the means. In addition to the means we must assume that a *duty* is involved given these two constants. We choose here the case of altruistic duty and introduce first duty as a constant and then a hypothesis

CONSTANT Duty : $G \rightarrow B \rightarrow \text{Prop}$.

HYPOTHESIS D-Obl : (Duty goods ben) .

The obligation that altruism is a duty given the means for altruistic actions can be formulated as a definition using LAMBDA for sake of clarity.

DEFINE OBL-ALT : $G \rightarrow B \rightarrow \text{Prop} :=$

[LAMBDA g :G; b : B]

((ALT-excl g b) \ / (EGO-moderate g b)) \ (D-Obl g b) .

Summary and Conclusion

Moral behaviour is modelled formally in LINCOS. Four kinds of moralism are distinguished by an inductive definition involving conditions. Three of them are grades of egoistic behaviour with altruistic aspects—the fourth is purely altruistic. The various kinds of moralism are shown to be deductible from the defining conditions.

The concepts and instruments utilized are those available in LINCOS, inspired by structures in the Calculus of Constructions with Induction (over types). This discussion is followed by a formalization of the concept of duty involving goods and benefits, the means for carrying out duties. Moral behaviour involving no benefit for self (pure altruism) or large benefit for others is considered to be an obligation.

Note that the formal LINCOS terms in the present chapter represent in fact extensive annotations to above summary. An extraterrestrial alien party receiving terms like these will certainly have difficulty with the interpretation. Verifications of facts (proofs) are not considered in detail in this chapter. They are interpretations of facts, but are not very helpful in communication via monologues with uninitiated extraterrestrial parties. If, however, at the receiving end it is recognized that the particular modality of logic, which we call strongly typed constructive, is utilized, then a transcription from our notational conventions to their representations is all what is needed to “read” the LINCOS texts. Fortunately LINCOS is capable of self-interpretation, see Chap 13 and reference (Ollongren 2003). Therefore the transcription task, hopefully leading to (partial) interpretation, should be a feasible one.

Notes, to begin with reminders, writing LAMBDA and ALL for sake of clarity:

1. Type is the basic type.

- CONSTANTS $D, R : \text{Type}$. is a declaration. D and R are global
- if D has a resident, say because CONSTANT $d : D$. was declared, then d justifies D , or alternatively D is verified by d ;
- $[\text{LAMBDA } x : D] \dots$, introduces the local variable x with as scope the body $\dots$;
- a mapping from domain D to range R is of type $D \rightarrow R$. For example if $r : R$ then $[\text{LAMBDA } x : D]r : D \rightarrow R$;
- $[\text{LAMBDA } a : D]a : D \rightarrow D$, the identity function
- $(\text{ALL } b : D)([\text{LAMBDA } a : D]a b) : D$.

2. CONSTANT Set : Type

CONSTANTS $G, B : \text{Set}$.

CONSTANT goods : G . CONSTANT ben : B .

CONSTANTS nil, moderate, large : B .

3. CONSTANT Prop : Type.

There are several admissible operations over the residents of Prop. We use the binary logical connective $\rightarrow$ (implication), but the binary $\wedge$ (and), $\vee$ (or), and the unary $\sim$ (not) are also available (see Chap. 2). The arrow $\rightarrow$ is evidently overloaded again.

Here then is the LINCOS model of moral behaviour separating clearly various cases:

4. CONSTANTS Act-by-person, Benefit-by-other : $G \rightarrow B \rightarrow \text{Prop}$.

CONSTANT Benefit-by-self : $B \rightarrow \text{Prop}$.

5. INDUCTIVE Moral-behaviour $[\text{LAMBDA } g:G; b:B] : \text{Prop} :=$

EGO-excl : $(\text{Act-by-person } g \ b) \rightarrow (\text{Ben-by-self } b) \rightarrow$

$(\text{Ben-by-other } g \ \text{nil}) \rightarrow$

$(\text{Moral-behaviour } g \ b) \mid$

```

EGO-large : (Act-by-person g b) → (Ben-by-self large)
→ (Ben-by-other g moderate) →
  (Moral-behaviour g b) |
EGO-moderate : (Act-by-person g b) → (Ben-by-self
moderate) → (Ben-by-other g large) →
  (Moral-behaviour g b) |
ALT-excl : (Act-by-person g b) → (Ben-by-self nil) →
  (Ben-by-other g b) →
  (Moral-behaviour g b).

```

The vertical stroke `|` serves to separate the four different cases. With above definition the type of Moral-behaviour is $G \rightarrow B \rightarrow \text{Prop}$. This is because LAMBDA introduces local variables (in this case g of type G and b of type B) which range over the body of the definition. The body is also of type $G \rightarrow B \rightarrow \text{Prop}$. Only one case represents altruistic moral behaviour.

6. The definition in the previous note implies for the occurring selector in the case of altruistic behaviour:

```

ALT-excl : (ALL g : G; b : B)
(Act-by-person g b) → (Ben-by-self nil) →
(Ben-by-other g b) →
(Moral-behaviour g b).

```

Given specific g and b we have

```

(ALT-excl g b) :
(Act-by-person g b) → (Ben-by-self nil) →
(Ben-by-other g b) →
(Moral-behaviour g b).

```

These formulations can easily be extended to all types of moralism within the context of the present paper.

```

7. INDUCTIVE Duty [LAMBDA g : G; b : B] : Prop :=
D-EGO-large:
  (ALL x : (Act-by-person g b); y : (Ben-by-self
large); z : (Ben-by-other g moderate))
  (EGO-large g b x y z) → (Duty g b) |
D-EGO-moderate:
  (ALL x : (Act-by-person g b); y : (Ben-by-self
moderate); z : (Ben-by-other g large))
  (EGO-moderate g b x y z) → (Duty g b) |
D-ALT-excl: (ALL x : (Act-by-person g b);
y : (Ben-by-self nil);
z : (Ben-by-other g b))
  (ALT-excl g b x y z) → (Duty g b).

```

References

- Douglas A. Vakoch and M. Matessa *An algorithmic approach to communicating reciprocal altruism in interstellar messages: drawing analogies between social and astrophysical phenomena*, Acta Astronautica 68 (2011), issues 3–4, 459–475
- A. Ollongren *Large-size Message Construction for ETI, Inductive Self-interpretation in LINCOS*, proceedings of IAC-03-IAA.9.2.09 (2003) contains an elementary introduction to the concept of induction and its use in definitions and applications

Part IV

Interpretation of LINCOS

Introduction

A message meant to be interpretable for ETI should evidently contain clues for its interpretation. If a human language is actually used as the main part of a message, there are formidable challenges to overcome at the receiving end because the syntax and semantics of the language employed must be completely unknown—see the long-standing problem of deciphering *Linear A*. The instance that there is no natural language contained in, or is not hidden in some way behind a message, might be less complex: assuming the abstract symbolism to contain (a sufficient number of) hints for its interpretation. The general case is that a message is composed of both comments on a topic in some language chosen by the sender and a set of formal descriptions regarding the topic. Note that these arguments apply as well the other way around: one might hope that a message from ETI meant to be interpretable for “others” (e.g., us in the case that the message is beamed in our direction) would contain a sufficient number of clues, internal consistency and redundancy, to enable interpretation by external intelligent beings (or information-processing machines).

Because of these observations it should be clear that a formal language for cosmic message construction (any *Lingua Cosmica*) should be “pure” in the sense that the possible sentences of the language are strictly defined syntactically by means of grammatical rules. The LINCOS system described in this book has a pure base derived from the calculus of constructions. A grain of impurity has been introduced in the elimination procedures discussed in Part I. These procedures are important and often necessary in argumentation. The present part sets out by “repairing” the formulated elimination function by reformulating it, and showing along the road that purely typed LINCOS is able to interpret itself.

In message construction, LINCOS should be used in juxtaposition with other means. In formulating a message for ETI using a (terrestrial) natural language in written form, a set of pictures (coded as bitmaps) illustrating the “story” told could be added—allowing interaction between text and pictures. In addition to this, messages might be augmented at another level (using a separate formalism) with descriptions of the semantics of certain parts of them—as mentioned before in this treatise. The selected parts can be sections, in some cases also individual declarative

sentences, but the expressions at another level are supposed to describe relations “living” in the text (a kind of story in general) transmitted. With examples we show in many places in the book how this can be achieved in practice.

Other means possibly helpful for interpretation are also discussed in the present part. Music could be part of a multilevel LINCOS message and we propose using Indonesian gamelan orchestral music for this purpose, i.e., the music itself together with the score. In that case the receiver would be enabled to “hear” the music and “read” the score. Scores are easily understood and it should be a relatively easy matter for “listeners” to correlate with one another a score and the distinctive sounds of individual instruments as the gongs.

The part ends with demonstration that the LINCOS system possesses a characteristic, distinctive signature, placing it in the realm of logic.

Chapter 13

Self-interpretation

“Hm, but you are hasty folk, I see,” said Treebeard. ... “I am not going to tell you my name, not yet at any rate.” ... “For one thing it would take a long while: my name is growing all the time, and I’ve lived a very long, long time; so my name is like a story. Real names tell you the story of the things they belong to in my language, in the Old Entish as you might say. It is a lovely language, but it takes a very long time to say anything in it, because we do not say anything in it, unless it is worth taking a long time to say, and to listen to.”

The Lord of the Rings, J.J.R. Tolkien, part II.

Caption

An old wild chestnut tree representing Treebeard (Fig. 4) . The tree with the author standing beneath it ‘in conversation with Treebeard’, was filmed by Prosper de Roos in 2008 for his international documentary ‘Calling E T’.

Intention

Even simple examples of texts may need extensive explanation, using LINCOS for formulating annotations—see Chap. 2. Generally the *elimination concept* is of prime importance in formulating verifications of a more involved nature in the annotations of the logic involved. A central position in this respect is occupied by the function ELIM, an unusual and rather elusive function. One might even argue that conceptually this function lies outside of “pure” typing. In Chap. 6 an alternative for carrying out certain eliminations (and circumventing ELIM) is already briefly mentioned. In the present chapter various alternatives are worked out in

Fig. 4 Treebeard

some detail. The setup discussed leads at the same time to the insight that the LINCOS system is capable of self-interpretation. By this we mean that the system, viewed as an operational system, is capable of achieving interpretation of expressions in terms of system concepts themselves without recourse to external means. This is achieved at the expense of in general sizable numbers of (often lengthy) formal terms. A crucial point in discussing the topic of self-interpretation in this context is the possibility of defining hypotheses.

Disposing of ELIM

In verification procedures eliminations are often involved and we have shown that the rather “strange” and in a way elusive function ELIM plays a role of prime importance, see Chaps. 2 and 5 but also Wittgenstein’s *Theatre*, Chap. 8. Since the function is typeless in a way, there is some reason to reconsider the use of it and instead

formulate eliminations strictly within “pure” LINCOS—thus in fact disposing of ELIM, see also the remark on this in Chap. 5. The effort yields a bonus: it leads to the insight that LINCOS is able to interpret itself. In a paper by the author and D. Vakoch (Ollongren and Vakoch 2002) this was pointed out for the first time. In (Ollongren 2003) inductive self-interpretation is placed in the limelight. The present chapter presents a different approach to the same topic.

The central idea of the present section is

- ELIM has the character of a universal function, but because it has no *bona fide* type one is interested in the possibility of replacing it by one or more specific functions tailored to specific goals
- In view these goals, specific hypotheses replacing ELIM are introduced.

Conjunction

Let A and B be two constants in the environment, as a result of declaring

```
CONSTANTS A, B : Prop.
```

We have seen in Chap. 2, writing $\wedge$ instead of and,

```
INDUCTIVE /\ [X, Y : Prop] : Prop :=
Conj : X → Y → (/\ X Y).
```

The two projection rules for conjunction mentioned in Chaps. 1 and 2 are

$(A \wedge B) \rightarrow A$ and $(A \wedge B) \rightarrow B$.

They can be verified as facts by using ELIM, but that is what we wish to avoid here. Consider, instead, the following hypothesis (because of its intended use called an inductive form)

```
HYPOTHESIS Elim-and : (X, Y, P : Prop) (X → Y → P)
→ (X /\ Y) → P.
```

Strictly speaking a hypothesis should only be entered in the environment after verification, in order to keep the environment sound (that is to say free of inconsistencies). That could be done in this case ... using the ELIM we wish to dispose of! We leave this aside and accept the hypothesis as such. In invoking the hypothesis at least three arguments are needed all of type Prop, for example A , B , and $A \wedge B$ in view of the first projection rule, because A is to be verified. We have then

$(\text{Elim-and } A \ B \ A) : (A \rightarrow B \rightarrow A) \rightarrow (A \wedge B) \rightarrow A$.

Had we chosen A , B , and B in view of the second projection rule, because B is to be verified, we find

$(\text{Elim-and } A \ B \ B) : (A \rightarrow B \rightarrow B) \rightarrow (A \wedge B) \rightarrow B$.

In order to verify the projection rules we need to construct two entities $p : (A \rightarrow B \rightarrow A)$ and $q : (A \rightarrow B \rightarrow B)$. That is done with the help of lambda forms as follows.

```
p = [h1 : A, h2 : B]h1 : (A → B → A) .
q = [h1 : A, h2 : B]h2 : (A → B → B) .
```

Note that what we need are the lambda forms, but not the individual p and q . All of this results in

```
(Elim-and A B A [h1 : A, h2 : B]h1) : (A /\ B) → A.
and (Elim-and A B B [h1 : A, h2 : B]h2) : (A /\ B) → B.
```

Suppose $a : A$ and $b : B$. One expects then that the entity $A \wedge B$ has a resident (is the case)—but that is not so. Instead we need an introduction rule, in the form of a hypothesis.

```
HYPOTHESIS intro-and : (X, Y : Prop) (X /\ Y) .
```

Disjunction

Again let A and B be two constants of type `Prop` in the environment as a result of a declaration. Supposing now the existence of $a : A$ one expects that the entity $A \vee B$ has a resident—and the same applies if $b : B$ is supposed to exist. Using the `ELIM` function that could be verified, but here we prefer to use two introduction rules for disjunction

```
A → (A \/ B) . and B → (A \/ B) .
```

They can be introduced as hypotheses (to be used as inductive forms) as follows

```
HYPOTHESIS intro-or1 : (X, Y : Prop) X → (X \/ Y) .
HYPOTHESIS intro-or2 : (X, Y : Prop) Y → (X \/ Y) .
```

Strictly speaking hypotheses should be verified first, as we emphasize repeatedly. That is done by recalling in this case the inductive definition of the disjunction from Chap. 2:

```
INDUCTIVE or[X, Y : Prop] : Prop :=
Prim : X → (or X Y) | Sec : Y → (or X Y) .
or : Prop → Prop → Prop.
Prim : (X, Y : Prop) X → (or X Y) .
Sec : (X, Y : Prop) Y → (or X Y) .
```

and renaming `Prim` by `intro-or1` and `Sec` by `intro-or2`.

Supposing $a : A$ and $b : B$ we find as expected

```
(intro-or1 A B a) : A \/ B.
(intro-or2 A B b) : A \/ B.
```

```
(intro-or1 B A b) : B \ / A.
(intro-or2 B A a) : B \ / A.
```

Consider the implication $(A \vee B) \rightarrow (B \vee A)$. Expressed as a fact, can this be verified?

```
FACT f.13.1 : (A \ / B )  $\rightarrow$  (B \ / A) .
```

Not surprisingly the verification is

```
f.13.1 = [h : (A \ / B)](intro-or1 B A b) :
(A \ / B )  $\rightarrow$  (B \ / A) .
```

Inductive Self-interpretation

An environment in LINCOS consisting of declared constants, variables, defined functions, and hypotheses, augmented with verified facts, can be considered to be a knowledge base. It can be extended not only by new declarations, definitions, and hypotheses but also by verifying new facts. The knowledge base is therefore non-static, and can grow on demand. Note that LINCOS does not supply instruments for deleting terms in the environment.

The system “knows” the information in the knowledge base and can use it. So we have in a way *per definition* self-interpretation. Consider, however, the above verification—it follows the general pattern of verifications. First the type of the fact is specified,

```
FACT f.13.1 : (A \ / B )  $\rightarrow$  (B \ / A) .
```

then it is followed by the constructed

```
f.13.1 = [h : (A \ / B)](intro-or1 B A b) .
```

Because $f.13.1$ and $[h : (A \vee B)](\text{intro-or1 } B \ A \ b)$ have the same type the fact is proved. For this verification the inductive definition of the disjunction function is needed, but

- No recourse is taken to information outside the knowledge base
- There is no reference to means external to LINCOS itself.

Above observation, present in various forms in several sections of this book, implies that the linguistic system is *closed* (free of references to external means of reasoning) and that it is able to explain itself within the system using an inductive principle.

The feature of (inductive) auto interpretation can be important in view of the question in which way an intelligent receiving system could be aided and supported in understanding some or most of the semantics of the LINCOS system. The present author does not suggest that receivers of messages should be provided with a

textbook describing LINCOS (in what language?) or the grammar of it in a perspicuous, understandable form. If a sufficiently large corpus of expressions is transmitted, augmented with a sufficiently large number of examples of the type shown above, an effective aid for “understanding” is provided. On the other hand, indeed we cannot hope that a complete understanding of the semantics and the mechanics of the system would be attainable, but that is perhaps not even necessary. Here on Earth, suppose we receive a message evidently emitted by ETI, we would be happy to be able, perhaps along the lines explained above, to understand some of the background of the (astro)linguistic system employed. This would hopefully yield a kind of base for further interpretation of message content.

References

- A. Ollongren and D. A. Vakoch *Large-Size Message Construction for ETI: Self-Interpretation in LINCOS*, International Astronomical Union, Symposium No. 213, Bioastronomy (2002): Life Among the Stars, p. 499–504
- A. Ollongren *Large-Size Message Construction for ETI: Inductive self-Interpretation in LINCOS*, Paper IAC-03-IAA.9.2.09, International Astronautical Congress (2003), Bremen

Chapter 14

Pictorial Representations

Intention

Suppose that we decide to include information in our messages to ETI in the form of pictures of various kinds. When choosing coding procedures, the designers of messages containing (digitised) pictorial information should keep an open mind for the serious decoding problem at the receiver's end—but we shall not be concerned especially with this aspect in the present chapter. Instead we address the question of what kind of information in connection with pictures one might select and choose to transmit. Once that has been decided, new LINCOS comes into play: pictures need formal annotations explaining the contents of the story told. At first sight this seems to imply restrictions on the kind of pictures to be considered. That is indeed the case, but even more important is the requirement that the story should possess “tangible” logic contents.

Pioneer Plaque

On each of the space probes Pioneer 10, launched in 1972, and Pioneer 11, launched in 1973, a golden plaque is attached to the frame of the antenna on the outside, somewhat shielded from impacts of interplanetary dust particles. The first probe did a flyby of Jupiter and continued out in the belt of asteroids, leaving the Solar system. Pioneer 11 did a flyby of Saturn and left the solar system as well via the asteroid belt. The (famous) plaque contains engraved pictorial information in various forms: a schematic outline of the probe itself, a spider-like figure showing in which directions 15 pulsars are seen from the Sun, together with a coding of their frequencies—and in addition line drawings of a woman and a man, both nude, drawn to size compared with the outline of the probe. It was the astrophysicist Carl Sagan's idea to include this information, inscribed in gold, on the antennas of the probes. The present author notes here that Carl Sagan and the radio-astronomer Frank Drake were pioneers in SETI. Many years before the launching of the probes, they had

been breaking ground in SETI-research. The artist Linda Salzman Carl Sagan's wife at the time did the design of the inscriptions.

The pictorial information on board the two Pioneers will certainly be enormously valuable and informative for the fortunate alien, intelligent being, picking up one of them. This is because the metric and time scales are immediately evident: the probe itself is there to show it. With a bit of luck the alien will also be able to ascertain the location of the sender Earth in the Galaxy, and gain an understanding of the epoch in galactic evolution in which the probe was launched.

This kind of pictorial information is, however, not of the kind the present author would preferably be interested to represent in LINCOS. For one thing the notion of scaling is foreign to the astrolinguistic system. The system does not refer directly to or is dependent of any scale of any kind outside or within itself. Secondly the specific parts of the astronomical world chosen for the Pioneer plaque (the pulsars and the constellation of the Sun and its planets) are not of prime interest in the discussions in this book. It would of course not be impossible to enter in a message, information in LINCOS on the 15 pulsars as we know them—and there is no real objection against doing so. But its usefulness would be limited from a linguistic point of view.

A Course in Latin

Many years ago The Nature Method Institutes represented in many European capital cities issued a written course LINGUA LATINA, secundum naturae rationem explicatae (Copenhagen 1959). The course contains explanations on the grammar and use of Latin, all of it explained in Latin itself—and it contains many line drawings, not many details in them. The way the course is constructed indicates a possible way of using pictorial information in messages for ETI. To give an idea of the contents of the introductory paragraph,

- The course starts with a *description* of Imperium Romanum, with its islands, rivers and cities, inclusive schematic maps.
- Then a Familia Romana, living in a house in Rome is introduced with drawings of the individual members: Iulius, (vir Romanus, pater est), Aemilia (femina Romana, est mater), Marcus et Quintus (pueri sunt) and Iulia (puella est). There are two servants: Medus servus Iulii est and Delia est ancilla Aemiliae (Rome in antiquity!).
- Individuals are distinguished from one another by their *names*.
- From there on an extensive discussion follows on various *relations* in the family, giving the student an idea of family life in ancient Rome. But this part is less interesting from the point of view of the present chapter.

The course in Latin contains pictures not only of individuals but also of events. Suppose the course would have been represented as a series of pictures in *comics*. Even then, with the pictures available, annotations in some linguistic

system would be useful and desirable. In comics each individual has a name attached to him/her/it, usually not written in a “balloon”, but somewhere in its vicinity. One of the main purposes of the balloons in comics is to explain intentions of individuals and happenings in the “scenes of action”. The occurring individuals have certain characteristics, widely different from one comic series to another. So if we wish to use a kind of comics and annotate stories in them, we should by using LINCOS be able *grosso modo* to describe at least the following:

- Individuals, their names, relations other individuals, family relations
- (Material) properties of individuals
- Locations of individuals
- Intentions of individuals, if possible
- Changes in the scene of events and or actions.

We show in the present chapter how to achieve in LINCOS descriptions of these kinds of relations. Partly they can be treated as explained in the Aristotelian Theatre (Chap. 7)—but we need additional means of expression—even though we restrict the discussion here to static logic relations. In order to discuss dynamic relations we need to take recourse to processes, treated further on, in Part V.

Individuals

In Chap. 3 we discussed a story from Alice in Wonderland and introduced Alice by the declaration

CONSTANT Alice : Prop.

and we introduced

CONSTANT what-you-were : Prop $\rightarrow$ Prop.

CONSTANT what-you-might-have-been : Prop $\rightarrow$ Prop.

This implies for instance

(what-you-were Alice) : Prop.

Suppose that we had introduced the Dutchess as well by the declaration

CONSTANT Dutchess: Prop.

In that case

(what-you-were Dutchess) $\wedge$ (what-you-might-have-been Dutchess) : Prop.

and there is *no difference* between Alice and the Dutchess as far as these two predications are concerned.

In Chap. 7 we remarked:

Aristotelian logic is unable to deal with single subjects (singulars) ... this has been considered to be a weakness. Fortunately the logic contents of sentences (... containing one or more singulars) can be expressed in the LINCOS system, even though individuals must be represented in a special way (so that their individuality is guaranteed).

We need a way of introducing this individual kind of subjects, the mentioned singulars. We choose as a useful example the plaques on the Pioneer probes, more in particular the line drawings of the two humans, a woman and a man. The pictures evidently represent two distinct figures, in fact distinct subjects. Let us name them Adam and Eva (what's in a name), or rather to say: we shall refer to them by these names. Their names are not on the plaque and there are no balloons. But the two figures evidently represent something distinct from the other (astronomical) information available. As there is no clothing drawn in the pictures it should be clear that a name given refers to a complete figure. Note that it is unclear which of the figures is named Adam and which Eva—but there is no need to clear up the ensuing ambiguity. The naming issue settled satisfactorily, we continue from here on with further more detailed characterisations.

To begin with we state that Eva is of type Female and Adam is of type Male; both females and males are of type Human. Note that we do need to specify the types of Human, Female and Male. In multilevel LINCOS we could provide further information of the meaning of these terms, but that is not very useful for the purpose of the present chapter.

CONSTANT : Human : Set.

CONSTANTS Female, Male : Human.

These type definitions must be augmented with definitions of two individual humans.

Eva as an individual is introduced with three definitions.

DEFINE is-individual-Eva : Female.

DEFINE Eva : is-individual-Eva.

DEFINE is-Eva : is-individual-Eva $\rightarrow$ Prop.

Eva as an individual is supposed to have a distinctive characterisation, expressed by the second definition. As a result (is-Eva Eva) : Prop, but (is-Eva x) is undefined for any x not Eva. In this way the individuality of Eva is guaranteed. For Adam we have in a similar way

DEFINE is-individual-Adam : Male.

DEFINE Adam : is-individual-Adam.

DEFINE is-Adam : is-individual-Adam $\rightarrow$ Prop.

Now Eva has a property in the sense of *being* somewhere: she is depicted on the Pioneer plaque, shortly: she *is* on the plaque. The notion of this property of an individual can be expressed like this:

DEFINE is-Eva-on-plaque : is-individual-Eva $\rightarrow$ Prop.

(is-Eva-on-plaque Eva) : Prop.

Suppose we wish to express the fact that Eva *and* Adam both are on the same plaque of the Pioneer probe. This could be done by a definition of is-Adam-on-plaque. Note, however, that in this example the conjunction *and* is explicitly used, so we have here a case similar to one of the books lying on a shelf, treated in Chap. 2. Of course books must be replaced by humans and shelf by plaque. Therefore there is no need for a separate declaration of is-Adam-on-plaque.

Apart from the fact that the two mentioned subjects are on the plaque, no special relation between the two individuals is expressed. Therefore no predications involving both of them need to be constructed.

Above shows how locations of objects in relation to other objects can be represented, let it be persons, houses, streets or cities, etc.

Intentions of Subjects

Another matter is the representation of intentions of individual subjects. An example in connection with Eva and Adam would be the hypothetical case that Linda Salzman engraved Eva first on the plaque with the intention of adding Adam to the picture later on (which she did of course). The realisation of Linda's intention is a *process*. The ingredients for formalising a process like this are described in PART V, notably via the notion of an arbiter.

As intentions of individuals (or groups) can vary as they are realised (or not), the representation of intentions (or developing intentions) is in general an extremely complex process. The same holds for changes in the scene of actions.

As far as comics are concerned the complexity results in the fact that usually a large number of pictures are needed to tell the story. LINCOS in the form presented in this book can be used for annotating "simple" pictures, but is not suitable for stories told with much pictorial information. For that purpose a new generation *Lingua Cosmica* would have to be developed.

Reference

NATURMETODENS SROGINSTITUT, Copenhagen (1959)

Chapter 15

Representation of Music

These gongs possess a sound that grips one through the splendour that emanates from them, spreading an atmosphere of truly restfulness and power...
(Kunst 1973)

Intention

Suppose that some ETI society receives a message from planet Earth part of it coded in LINCOS, but also containing music. We assume that at the receiving end it is understood (perhaps immediately) that the message contains of two separate levels and that one of them contains music. Learned beings of the receiving society concerned with the interpretation problem would probably be grateful if some kind of key were supplied in the message enabling them to understand the basics of the underlying logico-linguistic system used in the descriptive level. It is assumed that the music contained in the message is a piece of Indonesian Javanese *gamelan* music played on individual instruments or perhaps performed in an orchestral setting. The music is of course also digitised but is clearly distinct from the descriptive part. That part contains in fact annotations to the music, written in LINCOS.

The present chapter is based on an earlier unpublished essay by the author “The Cosmic Gamelan: Musical Tutorials for Interstellar Messages” for the workshop “The Art and Science of Interstellar Message Composition” held in Paris in 2002. A shortened version is in (Ollongren 2004).

Considerations

We have argued that the choice of logic as a common ground for astrolinguistic communication is motivated by the expectation that logic can be considered *ipso facto* to be a reasonable and useful common ground for interstellar communication between galactic symbolic species. One cannot expect a species without the power of logical reasoning to be able to interpret an interstellar meaningful message. It seems, however, not immediately clear how music can be helpful for the interpretation of messages. A common denominator of the application of logic is correct reasoning over abstractions of reality so that argumentation on the basis of tools of logic leads to reliable results. At the same

time logic is useful for clarification. These aspects are important but they do not occupy a central position in the present chapter—we take here an opposite view: how can music help intelligent beings to understand the structure and use of logico-linguistic LINCOS system. Music used here is not meant to be a means for artistic expression or enjoyment, but as an aid for the interpretation of logical terms. Why use Indonesian gamelan, as played on Java in Indonesia, for that purpose? To a large extent because of the relatively simple underlying musical concept. Gamelan music is based on either a five- or a seven-tone system, and scores are written *grosso modo* in linear notation, easily understandable. Music written in the classical chromatic scale (consisting of 12 tones in an octave), for instance, baroque music, could also be used, but then with scores in a sequential notation. Non-linear notation over groups of notes, e.g. trioles, ornamentations and syncopes, would in that case have to be rewritten in sequential form. For sake of simplicity this line of thought will not be pursued here.

Musical Units in Gamelan

A composition of Indonesian Javanese gamelan music, whether performed in an orchestral setting or for accompanying a shadow play, is opened by an introduction, the *bukâ*. This part consists of just one sequence (represented as integers) of drum beats carried out by the main player in an orchestra and leader of the ensemble. In this way an indication of the structure of the piece is given and the tempo is established. The drum beats are not toned.

As mentioned there are two scales of tones in gamelan music. In the pentatonic (five-tone) scale (*laras sléndro*), the notes are written –6, 1, 2, 3, 5, 6, 1+, where –6 and 6 as well as 1 and 1+ are an octave apart. The notes 1, 2, 3, 5, 6 are roughly evenly distributed over the octave. In the heptatonic (seven-tone) scale (*laras pélog*) the notes are written 1, 2, 3, 4, 5, 6, 7. In the 12-tone chromatic scale 1 corresponds roughly to d, 2 to d#, 3 to f, 4 lies between g and g#, 5 corresponds to a, 6 to a# and 7 lies between b and c [see the standard reference written by J. Kunst (Kunst 1973)]. These notes form the primary alphabet, or PA.

... Whoever has been fortunate enough, be it only once, to hear the benefaction of this timeless booming tone, dominating the teeming sounds of the gamelan, and to hear it as it were, come out of the silence of eternity, will forever carry it with him as a most precious memory:

Gong jumeplug mandul-mandul
Gumulung obaking waréh. [Javanese].

“The sound of the gong beaten heavily, rolls on its ponderous beats like the ocean tide”.

Musical units in gamelan music are of a certain logical type, and are said to reside in that type. Also in the case of music representation the type concept is used to express something particular about its residents. Thus the type Gam-u can be used to express that its residents are musical units in gamelan music. Gam-u itself resides in the universal type Set, which exists by virtue of an axiom. The semicolon is used for registering residence. So we write a declaration

CONSTANT Gam-u : Set.

Any basic unit of four gamelan tones (called a *gâtrâ* in gamelan music), e.g. the sequence of tones 2 1 2 6, is supposed to reside in Gam-u. A sequence of four basic tones like this one is written for sake of clarity as (2 1 2 6). Because of the basic character of *gâtrâ*'s, they can be considered to be constants. So we need declarations like

CONSTANT (2 1 2 6) : Gam-u.

CONSTANT (2 1 2 3) : Gam-u.

Declarations introduce building stones to be used in constructions. Suppose that the sequence of two *gâtrâ*'s (2 1 2 6) (2 1 2 3) is to be constructed. We could consider this sequence as a building block and declare it as a constant. But here we prefer to use a general method applicable to the construction of any sequence of *gâtrâ*'s¹. At the outset we shall consider any sequence of basic tones to be of type Gam-u.

In preparation for the construction consider $\text{Gam-u} \rightarrow \text{Gam-u}$. This is a new type representing a (as yet nameless) function (a mathematical one-to-one mapping) from Gam-u to itself. The function can be applied to one argument only, and with a reduction (or evaluation) mechanism a result of type Gam-u can be obtained. This type is not useful for the construction problem at hand because there are two building blocks, (2 1 2 6) and (2 1 2 3), given as arguments. Both are needed for generating the required sequence. So we introduce with a declaration another function, this time given *conc* (from the concept of concatenation) as its name, like this

CONSTANT *conc* : Gam-u $\rightarrow$ Gam-u $\rightarrow$ Gam-u.

This function needs two arguments and delivers upon reduction a result of type Gam-u. Given the arguments (2 1 2 6) and (2 1 2 3) in this order, the result should be the requested sequence. The following constructive definition achieves exactly that requirement

DEFINE *conc* : Gam-u $\rightarrow$ Gam-u $\rightarrow$ Gam-u :=
[LAMBDA x,y : Gam-u] x y.

Often the token LAMBDA is left out. Here in $x\ y : \text{Gam-u}$, $x\ y$ is a sequence, *not* the same as $(x\ y)$, which is functional application. The $:=$ symbol signifies that *conc* is identical to the LAMBDA expression followed by the sequence $x\ y$. They have the same type—strictly required for this kind of definition. So *conc* and [LAMBDA $x,y : \text{Gam-u}$] $x\ y$ are the same function. Both of them can be applied to two arguments, say a and b of type Gam-u. Under application the variables x and y , abstracted under LAMBDA, are bound to a and b and the sequence $a\ b$ replaces the sequence $x\ y$ (the regime operational here is called β -reduction). So

(*conc* (2 1 2 6) (2 1 2 3)) = (2 1 2 6) (2 1 2 3).

The constructed sequence of two *gâtrâ*'s is also a musical unit.

With the help of the function *conc* scores consisting of many sequences of basic units can be constructed. For example,

$$\begin{array}{c}
 (\text{conc } (\text{conc } (2 \ 1 \ 2 \ 6) (2 \ 1 \ 2 \ 3)) \ (5 \ 3 \ 2 \ 1) (3 \ 2 \ 3 \ 1)) = \\
 (2 \ 1 \ 2 \ 6) (2 \ 1 \ 2 \ 3) \\
 (5 \ 3 \ 2 \ 1) (3 \ 2 \ 3 \ 1)
 \end{array}$$

two lines of a well-known gamelan composition (see next section).

Gamelan Performance

A gamelan orchestra consists of a number of instruments. The main kinds are those that carry the main melody (*balungan* instruments) and those that mark the structure of the piece. In the *balungan* there are three types of *sarons* (instruments with bronze horizontal keys): the low tuned *saron demung*, the centrally tuned *saron barung* (usually three or four of those in an orchestra) and one smaller *saron peking*, tuned an octave higher and struck twice on every note, filling open spaces between the notes. The *balungan* also includes the *slenthem*, with bronze keys horizontally placed over broad, rather thin, resonating metal cylinders.

The structure of a piece is marked by beatings of the *kenong* (a bronze kettle), the *suwukan* (a middle-sized vertical gong), the *kempul* (a larger vertical gong) and the large vertically hung *gong ageng* (also called the *gong gedeh*). The latter gong has an extremely low sounding pitch. There is no conductor, but the *kendang* (drum) player directs the orchestra using a complicated score, not easily abstracted in LINCOS. We do not consider here the *bonang*, an instrument that supplies patterns of ornamentation to a melody. Other ornamentation is provided by the *siter* (a string instrument), the *suling* (a bamboo flute) and other less important instruments. There may be vocal parts but for the sake of clarity of interpretation these should not be included.

In the current section we will be concerned with detail structure in scores for gamelan music. Scores can be constructed with the method outlined in the previous section but we consider now only the results, i.e. scores themselves and aspects of performance. Neither generative procedures (rules for the creation of scores) nor the process of an orchestra performing a composition is described in logical terms in the present contribution. The reason for this is that we wish to use gamelan musical performances themselves to explain aspects of LINCOS to listeners.

A piece of music for gamelan orchestra consists of a score with a number of *gâtrâ*'s concatenated into lines and sections. Consider for instance the particular composition Ladrang Asmorondono written in *sléndro* (pentatonic scale). The score of the first section, the opening (usually designated by A.), consists of four lines

A. (2 1 2 6) (2 1 2 3')
 (5 3 2 1) (3 2 3 1')
 (6 3 2 1) (3 2 1 6')
 (5 3 2 1) (3 2 1 (6))

A *ladrang* is a kind of composition with certain specific rules on the number of basic musical units and their structure in the score, and is usually written in *sléndro*. The units are slightly different from those discussed until now. An underlined note signifies that the *kempul* is to be used, a quotation mark indicates that a *kenong* is to be sounded, and a pair of parenthesis enclosing a note means that sounding the *gong ageng* (or *gong gedeh*, the large gong) is to mark the end of the section. These instruments are used to mark the structure of the basic units and thus the composition. They are played together with the *balungan* instruments.

As mentioned PA is the type of gamelan notes. Let PA' be the type of those notes where the *kenong* is sounded, let PA- be the type of those notes where a *kempul* is heard, and let PA() be the type of those notes where the large gong resonates. A musical composition as a *ladrang* must fulfil structural criteria on the basic musical units, the *gâtrâ*'s. From the example above it is seen that there are four kinds of units: those with a *kenong* sounded, those with a *kempul* sounded, those without marking instrument and those where the *gong ageng* is heard. In order to be able to distinguish between these kinds, we use a constant declared like this

CONSTANT G : Gam-u $\rightarrow$ Prop.

This means that for instance (G (2 1 2 6)) : Prop, (G (2 1 2 3')) : Prop. Prop indicates the property that its residents are propositions, which obey the rules of the propositional calculus, i.e. compound expressions are of type Prop. As an example we have

(G (2 1 2 6)) : Prop.
 (G (2 1 2 3')) : Prop.
 (G (2 1 2 6)) /\ (G (2 1 2 3')) : Prop.

indicating that indeed (2 1 2 6) and (2 1 2 3') can be concatenated. This is consistent with

(G (2 1 2 6) (2 1 2 3')) : Prop.

In order to be able to check for instance whether a given *gâtrâ* has the correct structure we define the following functions.

DEFINE gâtrâ-O : PA $\rightarrow$ PA $\rightarrow$ PA $\rightarrow$ PA $\rightarrow$ Prop :=
 [LAMBDA u,v,w,x : PA] (G (u v w x)).
 DEFINE gâtrâ-kempul : PA $\rightarrow$ PA $\rightarrow$ PA $\rightarrow$ PA- $\rightarrow$ Prop :=
 [LAMBDA u,v,w : PA, x : PA-] (G (u v w x)).
 DEFINE gâtrâ -kenong : PA $\rightarrow$ PA $\rightarrow$ PA $\rightarrow$ PA' :=
 [LAMBDA u,v,w : PA, x : PA'] (G (u v w x)).
 DEFINE gâtrâ -gong-ageng : PA $\rightarrow$ PA $\rightarrow$ PA $\rightarrow$ PA() :=
 [LAMBDA u,v,w : PA, x : PA()] (G (u v w x)).

Examples of correct structures

```
(gâtrâ-O 2 1 2 6) : Prop
(gâtrâ-kempul 5 3 2 1-) : Prop
(gâtrâ -kenong 2 1 2 3') : Prop
(gâtrâ -gong-ageng 3 1 2 (6) ) : Prop
```

while, e.g. (gâtrâ-O 2 1 2 3'), (gâtrâ-kempul 2 1 2 3') and (gâtrâ -gong-ageng 2 1 2 3') are undefined.

Explaining LINCOS

In the previous sections of this chapter several aspects of LINCOS have been used. Could they be useful as a partial tutorial for the astrolinguistic system proposed? That can only be the case for humans familiar with concepts in Indonesian gamelan. Alternatively humans familiar with constructive logic can consider the discussion as an introduction to the theory and practice of gamelan playing!

For the purpose of explaining to ETI conventions of LINCOS and the way the language is used in the realm of message construction for interstellar communication, we could do the following. A structured message meant for an extraterrestrial intelligent society is constructed and transmitted as a modulated electromagnetic signal. It contains four blocks in some sequence. A text of considerable size in some natural language is in one block. In another block, annotations using LINCOS have been entered describing the logical contents of the text. The third block contains a suitable piece of gamelan music, and the fourth block comments (also in LINCOS) on the musical score, along the lines expounded in the previous sections of this chapter.

With the help of blocks three and four, extraterrestrial recipients are provided with possibilities of familiarising themselves with the conventions of LAMBDA and LINCOS. We assume, of course, that the concept of constructive logic is known and that recipients need only to be shown the notational conventions.

In addition to clarifying LAMBDA, the fourth block should also contain expressions of another kind: logical deductions. This inclusion would clarify some of the expressive power of constructive logic. The following examples clarify what we mean by this. In order to avoid unduly heavy formalism, they are not written in LINCOS, but are paraphrased in words.

In gamelan music there are several kinds of compositions.^{2,3} We have shown an example of a Ladrang, but there are several others, for instance the Lancaran. They are mutually distinguishable by the number and kind of gâtrâ's and especially by the way sequences are organised, i.e. the way the musical units are joined in composition. Also, each piece is performed in specific tempi. In general, the musical expressiveness is adapted to the occasion of the performance. We shall not be concerned with those aspects at the moment. From the discussion in the preceding sections, it should be clear that some of the structural aspects can be expressed in LINCOS, more or less along the lines in the following examples.

All opening sections of Ladrang's consist of four lines of two basic musical units (gâtrâ's) each. Each gâtrâ has a specific structure expressed by kempul, kenong and

gong ageng beats. Suppose a given score has eight basic musical units meeting the structural requirements in the right sequence. Then the section is that of a Ladrang.

All opening sections of Lancaran's consist of two identical lines of two or four gâtrâ's each. Each gâtrâ has a specific structure expressed by kempul, kenong and gong suwukan beats. Suppose a given score has four or eight musical units and that they meet the structural requirements in the right sequence. Then the opening is that of a Lancaran.

The criteria formulated above are purely formal and carry no claims of musicality whatsoever. In constructing an interstellar message along the lines proposed here, as much attention as possible should be given to the fact that the purpose of a gamelan performance, often in conjunction with shadow plays, is to tell a story. The performing orchestra characterises by musical means happenings in the stories and expresses emotions as well. In which way to use *Lingua Cosmica* for that purpose is another question altogether.

Notes

The author has played the large gongs in a gamelan orchestra for many years.

1. As sequences of gâtrâ's can be arbitrarily long, one needs the concept of (mathematical) induction to give a formal definition. In LINCOS inductive definitions are available.
2. Two examples of sections A. of Lancaran's written in the pélog scale:

Ricik-ricik (3 5' 6 5') (6 5' 7 (6))
(two identical lines)

Udan Mas (6 5 3 2') (6 5 3 2') (. 3 2 3') (6 5 3 (2))
(two identical lines)

In both cases the gong *suwukan* is used instead of the *gong ageng*. The dot means that no sound is produced.

3. A piece of music for a gamelan orchestra (*gendhing*) is composed according to quite a number of rules, and there exist several types. In addition to the *Ladrang* one can also use the important *Lancaran* type in interstellar message composition, because these two are rather easy to understand for listeners. The individual notes (tones) are easily discernible. Furthermore, there are many tempi at which a piece (*patet*) may be played: slow (*irama* 3 and 4, close to *largo*), lively (*irama* 1 and 2, close to *allegro*) or very lively (*patet manyuro*, close to *presto*). Which *patet* is used depends on the combination of tones in the piece, the purpose of the orchestra playing the piece, and whether a shadow play is being accompanied or whether a performance concertante is carried out. In practice excerpts of gamelan music in the *patets* slow or lively, using little or no ornamentation would meet our objectives best. It is hoped that listeners will then be able to distinguish the tones individually. There are of course different modalities in *sléndro* and in *pélog*, also in the *patet*, but we shall not be concerned with these refinements.

References

- J. Kunst *Music in Java, Its History, its Theory and its Technique*, Martinus Nijhoff, The Hague (1973), describes gamelan notes embedded in the 12-tone scale. In Vol. I on p.142 Kunst continues with:
- A. Ollongren *Music in Lingua Cosmica*, LEONARDO Vol. 37, No 1 (2004)

Chapter 16

Signature of LINCOS

Intention

Suppose the international SETI effort yields the discovery of some signal of evidently non-natural origin. Could it contain linguistic information formulated in some kind of *Lingua Cosmica*? One way to get insight into this matter is to consider whether specific (bio) linguistic *signatures* are likely to be attached to a cosmic language for interstellar communication—designed by humans or an alien society having reached a level of intelligence (and technology) comparable to or surpassing ours. How are they to be recognized and do they provide clues for the interpretation problem? In order to get some insight into this matter we consider a special characteristic of the logico-linguistic LINCOS system as described in the present book. The concept of *lambda abstraction* should qualify for that. It turns out that the characteristic can be used as a specific signature.

Introduction

The LINCOS system is *ohne weiteres* significantly distinct from natural languages because it has a specific twofold characteristic, unlike that of spoken or written languages on Earth. In fact *abstract* and *concrete* signatures can be distinguished. That an abstract kind occurs is due to the manner in which representations of aspects of reality are represented in the form of lambda (or λ) abstractions in LINCOS texts. The notion of λ -abstraction and the background of it is discussed in Appendix D. The representations can take compound forms because the system is multi-expressive—partly due to the availability of inductive (recursive) entities. On the other hand the distribution of delimiters and predefined tokens in texts is another characteristic of LINCOS, and indicates that the system has in addition a concrete signature as well. Assigning measures to concrete signatures will not be discussed here. The present chapter concentrates on the abstract signature of the language. At the same time it is realized that an alien *Lingua Cosmica* might but not necessarily need to have this

kind of signatures. *Note:* This chapter is based on a paper presented in 2008 by the author at the 1st International Astronautical Academy Symposium in Paris on Searching for Life Signatures (Ollongren 2008).

Subjects and Predications Revisited

As the LINCOS system is designed primarily to serve as an annotator for *logic* relations in texts describing aspects of reality, (symbolic) logic, amongst other ingredients has been embedded in LINCOS. As a generic case Aristotle's formalism is described in Chap. 7. So it should be no surprise that the basic constituents of the signature of LINCOS are *subjects* and *predications*. Both of these can explicitly be introduced as *types* by declarative means, i.e. in the forms of basic declarations and hypotheses—these forming the global environment (the universe of discourse). In LINCOS these entities (logic terms) are recognizable by their (concrete) declaratives as VARIABLE, CONSTANT, HYPOTHESIS, DEFINITION (sometimes written DEFINE), INDUCTIVE DEFINITION (usually written INDUCTIVE). In addition to global variables and constants, local variables implicitly bound to types can be introduced. In this case the binding (*typing*) of the variables, generally in definitions of maps, is by using the so-called typed λ -abstraction, but they can also be quantized, e.g., “for ALL x ”, or by the existence function EX. We emphasize that λ -abstractions themselves are typed as well. *Summa summarum* we find that this kind of typing is an important aspect of the abstract signature of LINCOS-texts.

Implication and Maps

In order to set the background for a “definition” of the signature of LINCOS, we review briefly some notation used in the system. At the same time the present chapter is supposed to be self-contained. So we are interested here in syntactic aspects and leave semantics aside.

Prop is a predefined type; it might but need not be interpreted as a set of properties. If (a property) A is of type Prop, written $A : \text{Prop}$, we have agreed that Prop is *inhabited* by A. Alternatively one can say that A is certain to *exist* (see Chap. 23). If (an object) a is of type A, A is inhabited by a and we say that A *is the case*. Let B be another property $B : \text{Prop}$. Then for the logical implication connective *imp*, usually written $\rightarrow$, we have seen using prefix notation

$$(\text{imp } A \ B) : \text{Prop}.$$

Terms over operators are written in *prefix notation*. The interpretation of above expression is that implication supplied with two arguments is a new property of type Prop, or in other words that the realm of properties is closed under implication. The

same applies to the other logical binary connectives, $\wedge$ (“and”), $\vee$ (“or”) and the unary $\sim$ (“negation”) over properties. Implication could be defined by

DEFINITION $\text{imp } [\lambda A, B : \text{Prop}] : \text{Prop} := A \rightarrow B \rightarrow \text{Prop}.$

Here $[\lambda A, B : \text{Prop}]$ is a typed λ -abstraction, introducing (in this case two) local entities of type Prop . The notation is varied from the λ -calculus in mathematics and logic (see Appendix D). In most chapters of this book the λ is omitted, but here we need to indicate explicitly when and where the λ is used. In the above definition $\rightarrow$ denotes a mapping (with the customary interpretation). It is the case that a mapping (simple or compound) possesses also a type.

Assuming the types $X, Y : \text{Prop}$, note the following:

$\text{imp} : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop}$
 $(\text{imp } X) : \text{Prop} \rightarrow \text{Prop}$
 $(\text{imp } X \ Y) : \text{Prop}.$

We could have written $\text{imp} : (\text{Prop} \rightarrow (\text{Prop} \rightarrow \text{Prop}))$ using the convention of brackets associating to the right in mappings.

For binary connectives we might use infix notation, for easy reading by humans. Implication occupies a central position in LINCOS. We have seen that the symbol $\rightarrow$, used in typing map definitions, is also used for logical implication, i.e. imp can be replaced by $\rightarrow$. Note, however, as we have seen in Chapter 2, that

DEFINITION $\rightarrow [\lambda A, B : \text{Prop}] : \text{Prop} := A \rightarrow B \rightarrow \text{Prop}.$

is unacceptable, even though it looks like *bona fide* recursive definition!

Suppose that $A \rightarrow B$ is the case and in addition that A is the case (in other words, it is given that for some a , A is the case). One expects then B to be the case. The fact

FACT $F1 : (A \rightarrow B) \rightarrow B.$

can be verified by constructing an expression for $F1$ of type $(A \rightarrow B) \rightarrow B$. Following the basic verification procedure in constructive logic we construct a λ -abstraction of the correct type. That is this one

$F1 = [\lambda H : A \rightarrow B] (H \ a).$

Here H of type $A \rightarrow B$ is abstracted, while $(H \ a)$ is the application of H to a . In order to determine the type of $F1$ represented by this expression, observe

$(H \ a) : B.$
 $F1 = [\lambda H : A \rightarrow B] (H \ a) : (A \rightarrow B) \rightarrow B. \text{ QED.}$

Subjects and Predications, an Example

Set is also a predefined type. It can but need not be interpreted as a general set (i.e. a collection of unspecified objects). If x is of type Set, written $x : \text{Set}$, we agree that Set is inhabited by (member) x . Sometimes this is expressed as follows: x resides in Set, or x exists, or (interestingly): x lives in Set.

Returning now to subjects S and predications P , we observe:

- If (an object) s is of type S , written $s : S$, S is inhabited by s .
- If (a property) p is of type P , i.e. $p : P$, we say that P is the case (P has property p might be used, but is avoided because this might give rise to confusion).

Given the type of subjects S , HYPOTHESIS $S : \text{Set}$
predications P over subjects are introduced as an unparametrized map

HYPOTHESIS $P : S \rightarrow \text{Prop}$.

We find then immediately for $s : S$, $(P s) : \text{Prop}$.

Let s be a particular inhabitant of S and suppose that a predication P is valid for all inhabitants of S , written $(\text{ALL } x : S)(P x)$. The symbol ALL is explicitly written in this chapter because it has to do with the signature of LINCOS. Then P must be valid for the particular inhabitant s of S , i.e. $(P s)$ should be the case.

FACT F2 : $(\text{ALL } x : S) (P x) \rightarrow (P s)$.

In order to verify this we need an expression for F2 of the type required. Following again the basic verification procedure in constructive logic we must construct a λ -abstraction of the correct type. Using here explicitly the token λ , we note here (with a hidden question mark) a suggestion:

$F2 = [\lambda H : (\text{ALL } x : S) (P x)] (H s)$.

In order to determine the type of this expression, observe

for any $Q : (P s)$ we have $[\lambda H : (\text{ALL } x : S) (P x)] Q : (\text{ALL } x : S) (P x) \rightarrow (P s)$.

Noting $(H s) : (P s)$, we find upon substituting $(H s)$ for Q

$F2 = [\lambda H : (\text{ALL } x : S) (P x)] (H s) : (\text{ALL } x : S) (P x) \rightarrow (P s)$. QED.

λ and the Signature

Above discussion reveals the important role of abstractions using λ , as they appear in global definitions in the environment and verification terms. In a given LINCOS text there may be many other ad hoc λ -abstractions, representations of something in reality. As explained the symbol λ is used whenever a type abstraction is in order.

Nesting of types is accepted (not shown in the examples). Local variables, introduced in this manner, have a restricted scope and must obey scope rules, but can be used again repeatedly elsewhere in λ -abstractions, thus much more freely than the global ones. One can also say that their lives are restricted to the bodies of the λ -abstractions but they can lead a second life elsewhere. Global variables range over large sections but cannot be redefined (a conflict between a local variable with the same name as a global one is resolved by scope rules of standard type).

Because of these aspects and the power of the kind of abstractions discussed, the significant λ -tokens together with the bodies involved, determine and govern *grosso modo* the abstract signature of LINCOS.

Next in significance is the semicolon-token in type designations. An important role is evidently played by the mapping operator $\rightarrow$ (not the implication connective) as well. Determining the correct interpretation of these symbols is of course far from trivial for an alien culture.

Note

The LINCOS terms occurring in above examples comprise hundreds of tokens. Of these about 5% are the colon punctuation mark, 2% are the symbol $\rightarrow$ and only about 1% are λ . The importance of the λ token is not due to the frequency of occurrence but because of the significant role of terms containing it.

Reference

Ollongren A (2008) *On the signature of LINCOS*, IAA - S8 – 0907, 1st International Astronautical Academy Symposium on Searching for Life Signatures, Paris, September 22–26, 2008

Part V

Processes in LINCOS

Introduction

This part of the book is concerned with ways and means of treating in a linguistic sense some dynamical aspects of reality—in fact the modeling of nonstatic relations in an astrolinguistic context. More in particular it shows how to incorporate and represent in LINCOS descriptions of (dynamical) processes without reference to time. The decision to disregard time was made because it is not at all evident how the concept of time can be explained within the framework of the *lingua cosmica*. The easiest kind of dynamical processes would be the case of sequential processes, of terminating and nonterminating sorts as they occur in various contexts on Earth, in everyday life. The more difficult ones are cooperating sequential processes, also common phenomena. And the very difficult ones are those in which information is transferred and passed along between various cooperating, concurrent processes. The participants in process activities, actors, be it humans or machines, will not be completely specified. There are a number of ways one can distinguish between the mentioned cases in an abstract sense: for example, those without or with arbiters, those interacting with signals by means of channels ... all of this is familiar and well known in processes occurring in daily life in human societies on Earth. However, modeling activities of these kinds, in the abstract setting “making ourselves understood for ETI,” is far from trivial. The following chapters treat some significant aspects of such an undertaking. The treatment is fundamental but does not aim to be complete.

The discourse in this part is based on aspects of Process Algebra as developed in computer science since the mid-nineties of the last century [1]. The central notion is that of a process, itself consisting of unspecified indivisible atomic actions—possibly of various kinds. Many process algebras have been developed—usually they are concerned with process behavior in time. As mentioned the concept of time is not present in the present treatment—instead sequences of processes are examined. Let p and q designate single processes. Different kinds of composite processes can be distinguished: p followed by termination, p followed by q , either p or q , p carried out in parallel with q , and other alternatives—for instance if there is an arbiter available. All of this is evidently not immediately present in the setup in the previous

chapters. We show in the present part how processes can neatly be incorporated in LINCOS, syntactically in an easy way. If necessary analyses of the behavior of processes can be done separately, using means outside of LINCOS.

Throughout the discussions processes are treated as abstractions—they are supposed to do something, but there are no specifications of the particular tasks to be carried out. Even in the case of the dining philosophers (Chap. 18), the processes of abstract thinking and concrete consuming (of liquids or whatever) are not specified. The relevant processes are supposed to be different from one another, mutually exclusive and noninterfering. An example: a philosopher cannot eat and think simultaneously!

Reference

1. *Handbook of Process Algebra* Edited by J.A. Bergstra, A. Ponse, A. Smolka, North-Holland, Elsevier (2001)

Chapter 17

Representing Processes

Intention

In this chapter we are concerned with the non-trivial matter of how to incorporate descriptions of sequences of *processes* without reference to time in *lingua cosmica*, the particular linguistic system for interstellar message construction proposed and discussed in detail in the present book. The chapter contains an important, fundamental extension of the discourse in the previous chapters in parts I–IV. Groundwork is carried out here for the purpose of interpreting (dynamic) processes of various sorts in the linguistic system. The concept of a process is given a prominent central position. At the basis we find elementary (or atomic, indivisible) processes. These can be combined into sequences of processes or concurrent processes. These are also processes, in fact almost everything is considered to be a process. But not all processes have the same type.

Introduction

A solid foundation is needed for the treatment in an astrolinguistic context of processes occurring in human (industrial) societies. For that purpose the concept of process needs to be abstracted in a logic sense. Processes are inductively defined in such a way that they can be arranged in sequences. There is no dependence on time. In the context of the present chapter it is relevant that processes are supposed to carry out actions, but there is no need to consider the internal constituents of them: these can be considered to be hidden. Therefore, processes can be abstracted and represented by variables. However, processes might have to communicate with the environment they reside in, or with other processes (for instance, because cooperation is called for). So we need ways and means to achieve that.

From the outset we stipulate that a process is associated with a *state* (the state of the process). Processes are provided with *channels*, mapping them to their states. The channels are considered to be processes as well. In this manner a base for

information exchange between processes and the representation of concurrent processes (next chapter) is provided. In the present chapter we discuss only static non-concurrent: processes, themselves *ipso facto* dynamic, arranged in some way that cannot be rearranged. In the next chapter we remove this restriction.

In order to model cooperating sequential processes, configuration information is usually needed. That kind of information can be registered in the form of *state vectors*. This suggests a special interpretation: a state vector, also a process, carries information, accessible to other processes, e.g. an *arbiter* (next chapter). Below we work out in somewhat detail some specific cases of cooperating sequential processes.

Sequences

We have seen that the central notion in using LINCOS is that terms (logical forms) are represented as *types* in the typed lambda calculus—this is also the case here when treating processes. So, to begin with a process is inductively defined as a sequential process or a channel process by

```
INDUCTIVE Proc := seq : Proc → Proc → Proc |
                  chan : Proc → State → Proc.
INDUCTIVE State := active : State | idling : State |
stop : State.
```

Elements from the so-called process algebra have influenced this choice. A channel process maps a process p to its unique state $(\text{chan } p) : \text{State}$. Note that two different processes can have the same state. They could both be active or idling, or doing the same thing. What an active process is supposed to be doing depends entirely on the situation the process is embedded in. It may be processing something, it may be passing information, a process may be changing something in its surroundings, or perhaps even creating something that was not there before. Whatever is the case, we allow the state of a process to be changed. Above definition reflects the basic assumption that there is a restricted set of states of processes, here: active, idling, and stop. Others can of course be introduced as required for applications. There is no ordering for these selectors.

If p , q and r , and q are of type Proc (they are considered to be processes), then $(\text{seq } p \ q)$ of type Proc represents the sequence of the two processes p — q (p followed by q) as a (sequential) process. Further $(\text{seq } (\text{seq } p \ q) \ r) : \text{Proc}$ represents the sequence of processes $((p$ — $q)$ — $r)$ as a process. In terms like this one, brackets associate to the left; therefore, we are allowed to write p — q — r . In this way sequential processes of any length can be represented in LINCOS. There is no reference to time.

Channels

Now consider channel processes denoted by `chan : Proc State Proc`. This type indicates something specific about processes. Let `p` be a process supposed not to be doing anything. Then

```
(chan p idling) : Proc.
```

is interpreted like this: the state of the process is idling—it is not communicating, it is doing nothing. A process like this is sometimes called a *silent* process. If `q` is a process supposed to be active we have `(chan q active) : Proc`. The processes `p` and `q` keep their states unless they are changed in some way.

Because `(chan p idling)` is a process, it can be an argument of `chan`. Using `chan` with such an argument can result in changes of states. Here are some examples of state changing. Starting with `(chan p idling) : Proc` we might have

```
(chan (chan p idling) active) : Proc.
```

The idling process `p` returns to active and as a side effect `(chan p active) : Proc`.

Suppose `(chan (chan p active) stop) : Proc`. Or `(chan (chan p idling) stop) : Proc`. Then the process *in fringe* terminates and as a side effect `(chan p stop) : Proc`.

Let there be, besides `p`, two other processes `q` and `r`. Assume `p` and `q` to be active and `r` to be idling. The following represents a sequence of the three processes:

```
(seq (seq (chan p active) (chan q active)) (chan r
idling)), abbreviated to
(chan p active)–(chan q active)–(chan r idling) :
Proc.
```

The example shows that the process `p` is active; it is followed by `q` also active and `q` is followed by `r`, idling. There is no information about possible relations between `p`, `q`, and `r`. Let this sequence be followed by

```
(chan p active)–(chan (chan q active) idling)–(chan
(chan r idling) active) : Proc.
```

in turn followed by

```
(chan (chan p active) idling)–(chan (chan q idling)
active)–(chan r active) : Proc.
```

In each sequence two processes are active and one is idling. If a process is idling, it will continue to do so until it is activated. If a process has stopped, it can be reactivated. An active process will stay active as long as it is not put in the state of idling or has terminated.

Example: Production of Strings of Symbols

Consider a sequence of processes, e.g. $p-s-q$. Available to p and q are a tape containing a linear sequence of symbols a , and another tape containing a stack of symbols 1 . The process p produces symbols a one after another on the tape, writing is done from left to right, and by concatenation a string is built on the tape. Each time an a is written, the symbol 1 is stacked (somewhere). Suppose p stops doing so after writing $n>0$ symbols. After that the silent process s does not do anything, but then q starts producing symbols b on the tape (they are concatenated to the string already on the tape) and removing the top of the stack each time a b is written. q stops when the stack is empty. It should be clear that the tape then contains a string of n times a followed by n times b , so it is the string $a^n b^n$. This is an example of a sequential process producing the so-called context-free language. If s is not silent and produces one symbol c on the tape concatenated to the string already there, the context-free language $a^n c b^n$ is produced. Strictly speaking the sequence $p-s-q$ should have n , the number of symbols to be processed, available as a parameter, so we have here a parametrization.

State Vectors

The state of a process can be set by using one of the following definitions, meant to activate the process.

```
DEFINE A : Proc → State := [x : Proc] active.
DEFINE I : Proc → State := [x : Proc] idling.
DEFINE S : Proc → State := [x : Proc] stop.
```

We have then for any $p : \text{Proc}$, $(A\ p)$, $(I\ p)$, and $(S\ p)$ are of type State . For sake of brevity we allow here: $(A\ p) = \text{active}$; $(I\ p) : \text{State}$ and $(I\ p) = \text{idle}$; and $(S\ p) : \text{State}$ and $(S\ p) = \text{stop}$.

Alternatively, in order to be able to retrieve the state of a process, we can use

```
DEFINE is-stop : Proc → State → Prop := [p : Proc; x : State] (= x stop).
DEFINE is-idling : Proc → State → Prop := [p : Proc; x : State] (= x idling).
DEFINE is-active : Proc → State → Prop := [p : Proc; x : State] (= x active).
```

Here equality is expressed with the $=$ token in the definition instead of using Eq .

Consider the sequence of processes, $p-s-q$, discussed in the previous section. Associated with this system is a state vector. The vector has at exactly three components. Initially p is active, and s and q are idling. So to begin with we have the state vector

```
(chan p (A p)) : Proc. (chan s (I s)) : Proc.
(chan q (I q)) : Proc.
```

Note that there is no information in this vector on the kind of action(s) the processes are carrying out—excepting the actions of s and q, who initially are doing nothing!

As the sequence of processes unfolds (taking n steps), the state vector is unchanged at first, but then it is followed by the vector

```
(chan p (I p)) : Proc. (chan s (A s)) : Proc.
(chan q idling) : Proc.
```

As the sequence continues to unfold (again n steps with unchanged state vector), it is followed by

```
(chan p idling) : Proc. (chan s (I s)) : Proc.
(chan q (A q)) : Proc.
```

The above is a comprehensive representation of stages of the sequential process $p \rightarrow s \rightarrow q$.

Note, however, that in this case first the processes s and q, then p and q, and finally p and s could be concurrent (silent) processes (processes running in parallel), because in the first case only p and in the second case only q is active. If it is required to register these remarks in some way we could write for the sequence of vectors (omitting the states)

$$p \mid \mid s \mid q \mid \mid \text{ or } p \mid \mid q \mid s \mid \mid, \quad s \mid \mid p \mid q \mid \mid \text{ or } s \mid \mid q \mid p \mid \mid$$

$$\text{and } q \mid \mid p \mid s \mid \mid \text{ or } q \mid \mid s \mid p \mid \mid.$$

In the next section we include the states by using subscripts.

Example: A Producer/Consumer System

Let P be a producer (serving products) and C a consumer (client) using data delivered by the producer. In the simplest case there is one bag B of unbounded capacity, in which P puts data, and from which the consumer takes data. The process P of producing data is represented by p. It is followed by b, placing the data in B. Both are active processes. This sequence, followed by c the act of consummation, yields a stable situation reached after some steps. So we have the sequence of state vectors

```
(chan p (I p)) (chan b (I b)) (chan c (I c))
(chan p (A p)) (chan b idling) (chan c idling)
(chan p active) (chan b (A b)) (chan c idling)
(chan p active)) (chan b active) (chan c (A c)),
```

or shortly

$$\mid p_i \mid b_i \mid c_i \mid \mid p_a \mid b_i \mid c_i \mid \mid p_a \mid b_a \mid c_i \mid \mid p_a \mid b_a \mid c_a \mid.$$

Here we have not taken into account the fact that some of these processes can occur in parallel. If the producer stops producing we have (representing is-stop by the subscript s)

$$|p_s \ b_a \ c_a| \ |p_s \ b_s \ c_a| \ |p_s \ b_s \ c_s|.$$

If the consumer stops consuming we have

$$|p_a \ b_a \ c_a| \ |p_a \ b_a \ c_s|,$$

but in this case there is no reason for the producer to stop producing, because the bag has unbounded capacity. So there is no need in this example to arrange for an escape clause for the case that the producer stops producing.

Example: A Producer/Client System

Let C be a client who puts requests (in the form of data) in bag B1. P is a producer who “reads” the data from B1, processes them, and returns the processed data in bag B2, which is then “read” by C. Both bags have unlimited capacity.

After putting a request in bag B1, the client C waits for a reply (the data are processed) from the producer P as deposited in bag B2. The producer in turn waits for a new request in bag B1. The producer “writes” or “reads” not both at the same time. The client either “requests” or “reads”, not both at the same time.

We distinguish eight processes:

c-out client C formulates a request and sends it to bag B1;
 b1-in bag B1 receives request; b1-out bag B1 passes request to producer P;
 p-in producer receives request; p-out producer produces reply;
 b2-in bag B2 receives reply; b2-out bag B2 passes reply to client;
 c-in client receives reply.

Using the shorthand introduced above, we write for the initial state vector

$$|c\text{-out}_i \ b1\text{-in}_i \ b1\text{-out}_i \ p\text{-in}_i \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_i|.$$

Suppose that a request is to be processed. Here is a possible sequence of state vectors as the request is handled by the system:

$$\begin{array}{l} |c\text{-out}_a \ b1\text{-in}_i \ b1\text{-out}_i \ p\text{-in}_i \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_i|. \\ |c\text{-out}_i \ b1\text{-in}_a \ b1\text{-out}_i \ p\text{-in}_i \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_i|. \\ |c\text{-out}_i \ b1\text{-in}_i \ b1\text{-out}_a \ p\text{-in}_i \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_i|. \\ |c\text{-out}_i \ b1\text{-in}_i \ b1\text{-out}_i \ p\text{-in}_a \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_i|. \\ |c\text{-out}_i \ b1\text{-in}_i \ b1\text{-out}_i \ p\text{-in}_i \ p\text{-out}_i \ b2\text{-in}_i \ b2\text{-out}_i \ c\text{-in}_a|. \end{array}$$

The next state is

$$|c-out_i \ b1-in_i \ b1-out_i \ p-in_i \ p-out_i \ b2-in_i \ b2-out_i \\ c-in_i|.$$

If a new request arrives, the handling represented above starts all over again. If not, the system keeps on idling, because we do not have a “shut off” state.

In the case described above there is information exchange between processes—represented by the travelling of a request (information) from one process to another. In the case described exactly one request travels through the sequence of state vectors and most of the processes are idling. The processing is done sequentially because a request follows a fixed order as it is handled. The bags have unlimited capacity but there is not more than one request at a time in bag B1 and a processed request in bag B2—or being processed in any other process. That means that a strict kind of cooperation between the processes is enforced in this example—in fact as a result of the way a requests (or requests) are handled (see the note below). In general producer/consumer systems handle several requests simultaneously, while the need for cooperation is maintained. The next chapter introduces the notion of cooperation in systems of parallel processes. Note that in above case the processes in each of the state vectors can very well run concurrently.

Note

The “travelling” of information between processes in the producer/client system can be illustrated using a binary code like this: 1000–0100–0010–0001. Much more realistic and useful in a way is the case coded like this: 1000–1100–1110–1111.

Chapter 18

Cooperating Sequential Processes

Intention

The present chapter is concerned with the non-trivial matter of how to incorporate in LINCOS descriptions of two or more *cooperating concurrent* sequential processes without reference to time. For that purpose the concept of *arbitration* is introduced. The arbiter is again a (typed) process, albeit of a special character. It has the power of organizing processes in such a way that by means of interrupts (with or without returns), a wide range of requirements of cooperation are met. As an illustrative example the well-known problem of organizing the concurrent dining and studying of an uneven number of dining/thinking philosophers is discussed.

Introduction

In the previous chapter we have encountered the situation that in modelling processes in LINCOS some cases may call for rearrangement of sequences of processes. In view of the basic assumptions formulated in the chapter, we conclude that a rearrangement can only be effectuated by changing the state vector of a sequence of (elementary) processes. We shall use an arbiter for that purpose, worked out in some detail below. Because of that and because we wish to discuss the treatment of concurrent processes, we must supply an extension of the inductive definition of a process supplied in Chap. 17. The definition admits descriptions of properties of processes arranged in sequences. Strictly speaking concurrent processes are outside the realm of the discourse in the chapter. Yet we have already seen that in state vectors channel representations can occur of essentially concurrent processes.

Concurrency

In order to render the concept of concurrent processes official we rewrite the inductive definition of a process as follows, introducing parallel processes:

```

INDUCTIVE Proc := seq : Proc → Proc → Proc |
                  par : Proc → Proc → Proc |
                  chan : Proc → State → Proc |
                  arb : Proc → State → Proc → State →
                  Proc.
INDUCTIVE State := stop : State | idling : State |
active : State.

```

It would seem that the introduction of the selector `par` is purely syntactical. Suppose `p` and `q` are two processes. Then `p-q` represents a sequence as we have seen, but (as we shall see in the next chapter) `||p q||` expresses that `p` and `q` are concurrent.

A syntactical definition of an arbiter (selected by `arb`) is included. As far as semantics are concerned, it should accommodate the notion of interrupting a sequential process and replacing it by another one—with or without arranging for a return to the interrupted situation.

Arbitration

The intention of arbitration as formally introduced above is the effectuation of an interruption of a sequence of processes, and letting it be continued by another sequence. Let `p` and `q` be processes and `s` be a state. Then by way of examples, starting with `(arb p * s active) : Proc` and `(chan s active) : Proc`, we define the equalities

```

(arb p * s active) = (chan s active) : Proc
or (arb p * s idle) = (chan s idle) : Proc

```

for another possibility. As before `*` represents: it “doesn’t matter” what argument is used here, provided that it is a state. Strictly speaking we tread outside the boundaries of formal LINCOS with these equalities. Repairing this digression would lead the discussion too much astray.

Arranged here is that whatever the state of the process `p` is (expressed by `*`), the arbiter replaces it by the process `(chan s active)`, and in the second example the `(chan s idle)`. So `s` is introduced at the expense of `p`. In this manner in a sequence of active processes ... `p`, `p` is replaced by `s`. A return can be realized by `(arb s * p *)`, supposing the state of `p` to be known before the interrupt. Two other examples are

```

(arb p * p idle) = (chan p idle).
(arb p * p active) = (chan p active).

```

Whatever the state of p is, in the examples it is first set to idle by the arbiter and then set to active.

A sequential process is associated with a state vector, also a process. Therefore, with the help of the arbiter, a sequential process can be interrupted and replaced by another process by changing the current state vector. We shall see that the same applies for concurrent (parallel) processes.

Five Dining Philosophers

As a representative example of replacements of state vectors consider a simple situation of limited resources and resource-demands, where arbitration is called for to ensure a fair kind of distribution between the process actions and available resources. The way how arbitration can be utilized for this purpose is described in this section. The following situation, rather famous in informatics, is provided with a model.

A round table is available for an uneven number n of philosophers to dine. On the table are n plates placed, and n forks: each plate has one fork to the right and one fork to the left of it. A philosopher sitting at the table needs both forks to be able to dine. A philosopher is either thinking (process t) or dining (process d), not both at the same time. Dining and thinking are done in a finite fixed number of steps. Steps are not related to time, and the interpretation is left open. Evidently only $(n-1)/2$ philosophers can dine simultaneously. This is because they need $((n-1)/2)2 + 1 = n$ forks for dining.

We choose for this example $n=5$. Let us name the philosophers P1, P2, P3, P4 and P5 and let the initial situation be that philosophers P1 and P3 are dining, the others are thinking (of philosophy). The process of dining is represented by *active*, and the process of thinking somewhat disrespectfully by *idle*. There are initially two dining processes, say $p1$ and $p3$, and three thinking processes $p2$, $p4$ and $p5$. Note that these five processes run concurrently. The initial state vector of the processes is then

```
|(chan p1 active) (chan p2 idle) (chan p3 active)
(chan p4 idle) (chan p5 idle)||.
```

Suppose that philosophers P1 and P3 leave the table in order to return to philosophizing, while two of the others decide to go dining. That means that either P2 and P4 or P5 and P2 could go dining. Which combination to choose? Let us say that the configuration is to be changed into the state vector

```
|(chan p1 idle) (chan p2 active) (chan p3 idle)
(chan p4 active) (chan p5 idle)||.
```

The arbiter is used for that

```
(arb p1 * p1 idle) = (chan p1 idle).
(arb p2 * p2 active) = (chan p2 active).
```

```
(arb p3 * p3 idle) = (chan p3 idle) .
(arb p4 * p4 active) = (chan p4 active) .
```

The (arbitration) sequence is a sequential process and cannot be considered to be a parallel process. So we can write

```
(chan p1 idle) - (chan p2 active) - (chan p3 idle)
- (chan p4 active) .
```

What matters here is that arbitration results in the required state vector.

Above is just one step in the arrangements needed to insure that a fair regime is attained for all philosophers spending equal number of steps thinking and equal number of steps dining. At the same time “collisions of interest” must be excluded: e.g. the case of two philosophers wanting to dine while there is only one set of forks (or even none) available at that moment.

So we are interested in a sequence of state vectors representing indeed a fair distribution of process actions, given available resources. Above is shown how part of this requirement can be realized. Next we look at the general case of the five philosophers.

Using a shorthand notation (similar to the one in the Note in Chap. 17) the first configuration will be written $\|1\ 0\ 1\ 0\ 0\|$ and the second $\|0\ 1\ 0\ 1\ 0\|$, thus expressing that we have parallel processes. After a while (a finite number of discrete steps in some arbitrary step counting unit) a new configuration and state vector of processes will occur: some philosophers will leave the table, others will stop thinking and wish to dine. We choose the following regime, a sequence of state vectors

```
| | 1 0 1 0 0 | |
| | 0 1 0 1 0 | |
| | 0 0 1 0 1 | |
| | 1 0 0 1 0 | |
| | 0 1 0 0 1 | |
```

representing the five possible situations. This sequence can be considered to be a kind of program written in a code. Interpreting the program as was done for one step above means that required sequences of state vectors meeting given demands are obtained.

Note that a program like this sequence or somehow based on it does not exist as a type within the apparatus of LINCOS—in fact one would need another level for reasoning about programs of this kind.

Conclusion

In the example of the five dining philosophers the specific choice of state process vectors and channels shows how an arbiter can be used to ensure a fair distribution of process demands given a limited number of resources. In this particular case the

arbiter together with a program arranges an acceptable series of process actions satisfying the constraints and meeting the fairness criterion.

In the basic set-up (the initial environment) two types of processes are introduced: sequential processes and processes of arbitration (associated with seq and arb), but the processes themselves are not detailed in any way—apart from the requirement that state vectors are associated with them. In addition there is no reference to time or the number of steps executing the processes required. Also the meaning of a step is left open.

Could the case of the five dining philosophers be extended to the case of an odd number of diners larger than five? One would have to introduce an *inductive principle* for that purpose. To begin with: Given the case of n (odd) diners, how is the program extended to the case of $n+2$ diners? Mathematically speaking this matter is trivially solved and an embedding of the principle in constructive logic is certainly possible—but the enterprise would lead us astray from the purposes of the present book.

Whether or not a program of arbitration specifies an acceptable series of actions for a given purpose is a matter for discussion at a higher level, in fact a *meta level*, where methods from mathematics or logic can be used for expressing statements and proving theorems. In the case described above, elementary statistical analysis shows that the chosen distribution program together with a suitable rule for sequencing yields fairness.

References

- A. Ollongren and D. Vakoch *Processes in Lingua Cosmica*, in ‘Communication with ExtraTerrestrial Intelligence’ (Edited by Douglas A. Vakoch, Suny Press, State University New York), AbSciCon 2010, Texas USA, Session ‘Search for Intelligent Life’, topic ‘Interstellar Message Construction: Can We Make Ourselves Understood?’, (2011) 413–418
- A. Ollongren *Processes in Lingua Cosmica* (source Shakespeare’s Hamlet), Acta Astronautica 71 (2012), 170–172

Chapter 19

Hamlet in LINCOS

Intention

As demonstrated in the previous chapter the concept of a process, elementary or composite can be abstracted in a logic sense and provided with basic properties. In this chapter *Proc*, the type process, is to be used as the type of parallel and arbitration processes. So here we are mainly interested in concurrent processes. Channels are introduced as a means to represent processes in *state vectors*. These contain representations of active processes. State vector transitions model *communication* between processes (locally or globally). Arbitration is represented by *programs* describing vector transitions. An example from Shakespeare's Hamlet illustrates the way this is effectuated in practice.

Introduction and Basics

We are interested in the present chapter in describing in LINCOS some “concrete” examples of processes running in parallel without reference to time. So we use only a part of the introductory map for *Proc* in Chap. 18 by defining only parallel processes and separately another kind of arbitration function.

```
INDUCTIVE Proc := par : Proc → Proc → Proc .
DEFINE arb : Proc → Proc → Proc :=
[x : Proc; y : Proc] ||y|| .
```

The meaning of $||y||$ is as follows: The process y , part of a concurrent system of processes (see next section), has *state* $||y||$, also a process. Because a process can be composite, there is no summing up of possible individual states as in the previous chapter. There is no need for ordering of processes, all of type *Proc*, occurring in parallel. If p is a process,

$(\text{par } p) : \text{Proc} \rightarrow \text{Proc}$, $(\text{par } p)$ can be applied to a process $a : \text{Proc}$ if and only if a does not interfere with p . For $((\text{par } p) a)$ we write $(\text{par } p a)$ or $(\text{par } a p)$.

In addition to these preliminaries we require: if p and q are processes, both *do* something but they are supposed *not* to interfere with one another. It is also supposed that there is no (operational) difference between $(\text{par } p \ q)$ and $(\text{par } q \ p)$. Any process, also one occurring in parallel with another, can terminate. Finally we stipulate that no process can occur in parallel with itself.

The arbitration process is included for the purpose of arranging sequences of processes, replacing processes by other ones, etc., subjected to a communication program to be detailed further on.

Parallel Processes

Let $p : \text{Proc}$ and $q : \text{Proc}$ be two unequal (possibly elementary) processes, which do something simultaneously (in parallel) without interfering with one another. We have then $(\text{par } p \ q) : \text{Proc}$. In Chap. 18 we used a small number of states of processes, but in this chapter we leave this open. We shall be writing for a pair like $(\text{par } p \ q)$ the expression $\|p \ q\|$ or $(\|p \ q\|)$ as a shorthand for $(\text{par } p \ q)$, but also as an element of a *state vector* of a special kind for these processes (elaborated further on in the present section). A disadvantage of this representation is that there is seemingly loss of information because the states are not formally defined. So the term *state vector* is in this stage almost a misnomer. When necessary, however, state information can be obtained explicitly using the selector *chan* as defined in the previous chapter.

We agree on the following: Let p be a process. Then $\|p\|$ representing p is available in the state vector. This representation can be considered to be a *silent* process of type Proc . So $\|p\| : \text{Proc}$. The following definition maps a process to its element in the state vector:

DEFINE $\text{channel} : \text{Proc} \rightarrow \text{Proc} := [x : \text{Proc}] \mid \|x\| \mid$.

Here $[x : \text{Proc}] \|x\|$ means that x is λ -bound to the type Proc , and the λ -term is applied to $\|x\|$. The resulting term is of type $\text{Proc} \rightarrow \text{Proc}$. Above definition is supplemented by a *revert*, mapping a vector element to the corresponding process.

DEFINE $\text{revert} : \text{Proc} \rightarrow \text{Proc} := [\mid \|x\| \mid : \text{Proc}] \ x$.

Here $[\mid \|x\| \mid : \text{Proc}] \ x$ means that $\|x\|$ is λ -bound to the type Proc , and the λ -term is applied to x . The resulting term is of type $\text{Proc} \rightarrow \text{Proc}$. If $p : \text{Proc}$ we agree on the conventions $(\text{channel } p) = \|p\| : \text{Proc}$, $(\text{revert } \|p\|) = p : \text{Proc}$ and find then

$(\text{revert } (\text{channel } p)) = p$ and $(\text{channel } (\text{revert } \mid \|p\| \mid))$
 $= \mid \|p\| \mid$.

Strictly speaking the so-called β -conversion of the λ -calculus must be used for this (see Appendix D).

In the case of two parallel processes p and q , $(\text{par } p \ q)$ can explicitly be mapped to $\llbracket p \ q \rrbracket$ by applying the channel to it, so we write $(\text{channel } (\text{par } p \ q)) = \llbracket p \ q \rrbracket$. The vector element $\llbracket p \ q \rrbracket$ is transformed back to $(\text{par } p \ q)$ by the application

$$(\text{revert } \llbracket p \ q \rrbracket) = (\text{par } p \ q).$$

The state vector is conveniently considered in this chapter to consist exclusively of (in a way elementary) silent processes of type Proc . Each of these corresponds to a “real” process. Note that the silent process $\llbracket p \ q \rrbracket : \text{Proc}$ is the same as $\llbracket q \ p \rrbracket : \text{Proc}$. In other words, $\llbracket \cdot \cdot \rrbracket$ is invariant under permutations of the (silent) processes. A silent process does not actually *do* something, but it carries information. Furthermore we assume that to each sequence of (parallel) processes a separate element of the state vector is made available.

The state vector of four processes, p, q, r, s , arranged like this

$(\text{par } p \ (\text{par } q \ (\text{par } r \ s)))$, or in some other permutation, is then $\llbracket p \ q \ r \ s \rrbracket$. This result is obtained by repeated application of channel. This four tuple is by convention also considered to be a silent process of type Proc . We require

$$(\text{revert } \llbracket p \ q \ r \ s \rrbracket) = (\text{par } p \ (\text{par } q \ (\text{par } r \ s))).$$

For the sake of simplicity we allow only elementary processes within two tokens $\llbracket \cdot \rrbracket$. This is chosen to avoid nesting: state vectors within state vectors. The representations explained have been chosen in order to admit the modelling in LINCOS of interrupts and returns in processes. This is achieved by means of an arbiter program.

The Arbiter for Parallel Processes

The purpose of the arbiter, the type of which is defined in the previous section, is to organize processing in a predetermined way using channel information. That goal is achieved by using the arbiter in a program. In addition to the definitions of channel and return, we use the type of arb given above. The typing expresses the intention of switching from one sequence of concurrent processes to another, e.g. from $\llbracket p \ q \rrbracket$ to $\llbracket r \ s \rrbracket$, where p, q, r and s are elementary.

Here are some examples, remembering

```

DEFINE arb : Proc → Proc → Proc :=
[x : Proc; y : Proc]  $\llbracket y \rrbracket$ .

(arb  $\llbracket p \ q \rrbracket$ ) : Proc → Proc. (arb  $\llbracket p \ q \rrbracket$ ) =
[y : Proc]  $\llbracket y \rrbracket$ .

(arb  $\llbracket p \ q \rrbracket \llbracket r \ s \rrbracket$ ) : Proc. (arb  $\llbracket p \ q \rrbracket \llbracket r \ s \rrbracket$ ) =
 $\llbracket r \ s \rrbracket$ .

```

Again, strictly speaking we tread a little beyond the boundaries of formal LINCOS with equalities like these. This is acceptable for the sake of clarity. Returning to $\llbracket p \ q \rrbracket$ after the second switch is effectuated by $(\text{arb } \llbracket r \ s \rrbracket \llbracket p \ q \rrbracket) = \llbracket p \ q \rrbracket : \text{Proc}$.

Application of arbitration is done sequentially. The argument of the arbiter is usually one element of a state vector, but the number of atomic processes in the element is unbounded. So it is possible to enlarge the number of processes under arbitration, but also to diminish the number. This means that care must be taken not to loose processes as a state vector unfolds.

The arbiter can be used to organize concurrent processing in a predetermined way. That goal is achieved by using the arbiter in communication *programs*.

As an example consider two programs.

```
DEFINE P1 : Proc → Proc := [| | x | | : Proc]
(arb | | x | | | | r s | |).
```

$\| x \|$ is λ -bound to the type Proc. If $a : \text{Proc}$ then (channel a) = $\| a \| : \text{Proc}$ and

$(P1 \ | \ | a \ | \ |) = (arb \ | \ | a \ | \ | \ | r \ s \ | \ |) = \ | \ | r \ s \ | \ |$.

Note that $\| x \|$ is variable, and $\| r \ s \|$ is a constant.

In this example the process a with silent state vector $\| a \|$ is interrupted and replaced by the process with silent state vector $\| r \ s \|$. Note: Here again β -reduction from the λ -calculus is applied. If a return to the constant $\| p \ q \|$ is required we use another program.

```
DEFINE P2 : Proc → Proc := [| | x | | : Proc]
(arb | | x | | | | p q | |).
```

So that for process with state vector $\| r \ s \|$ we have

$(P2 \ | \ | r \ s \ | \ |) = (arb \ | \ | r \ s \ | \ | \ | p \ q \ | \ |) = \ | \ | p \ q \ | \ |$.

A program is used to change an element of elements of a state vector.

Application, Opening Act in Hamlet

Using the concepts explained above we show now how actions in “reality” can be modelled in LINCOS. As an example we consider the processes occurring in the opening act of Shakespeare’s HAMLET, PRINCE OF DENMARK, SCENE,-- ELSINORE

ACT I. *A platform before the Castle.*

After a while there are four persons on the platform: FRANSISCO, BERNARDO, HORATIO and MARCELLUS. They are engaged in conversation when the GHOST enters.

Here is Shakespeare’s text leading to that event:

FRANSISCO *at his post*. Enter to him BERNARDO.

Ber. Who’s there?

Fran. Nay, answer me: stand and unfold Yourself.

Ber. Long live the king!

Fran. Bernardo?

Ber. He. *Fran.* You come most carefully upon your hour.

Ber. 'Tis now struck twelve; get thee to bed, Francisco.

Fran. For this relief much thanks: 'tis bitter cold. And I am sick at heart.

Ber. Have you had quiet guard?

Fran. Not a mouse stirring.

Ber. Well, good night. If you do meet Horatio and Marcellus, The rivals of my watch,
bid them make haste.

Fran. I think I hear them. – Stand, ho! Who is there?

Enter HORATIO and MARCELLUS. Friends to this ground.

Mar. And liegemen to the Dane.

Fran. Give you good-night.

Mar. O, farewell, honest soldier: Who hath reliev'd you?

Fran. Bernardo has my place. Give you good-night. [*Exit.*]

Mar. Holla! Bernardo!

Ber. Say. What, is Horatio there?

Hor. A piece of him.

Ber. Welcome, Horatio:- welcome, good Marcellus.

Mar. What, has this thing appear'd again to-night?

Ber. I have seen nothing.

Mar. Peace, break thee off; look where it comes again!

Enter GHOST, armed.

We use the following abbreviations for the elements in state vectors:

Fran-p = *Fran* on platform, *Fran-c* = *Fran* conversing,
Fran-ex = *Fran* leaving

Ber-e = *Ber* entering, *Ber-p* = *Ber* on platform, *Ber-c*
= *Ber* conversing

Hor-e = *Hor* entering, *Hor-p* = *Hor* on platform, *Hor-c*
= *Hor* conversing

Mar-e = *Mar* entering, *Mar-p* = *Mar* on platform, *Mar-c*
= *Mar* conversing,

Mar-! = *Mar* exclaims

Ghost-e = *Ghost* entering

In the beginning FRANSISCO is on the platform; thus the state vector is then
|| *Fran-p* ||.

The developing sequence of state vectors is during the beginning of Act I:

```

|| Fran-p ||
|| Fran-p Ber-e ||
|| Fran-p Ber-p ||
|| Fran-p Fran-c Ber-p Ber-c ||
|| Fran-p Fran-c Ber-p Ber-c Hor-e ||
|| Fran-p Fran-c Ber-p Ber-c Hor-p Mar-e ||
|| Fran-p Fran-c Ber-p Ber-c Hor-p Mar-p ||
|| Fran-p Fran-c Ber-p Ber-c Hor-p Hor-c Mar-p Mar-c ||
|| Fran-ex Ber-p Hor-p Mar-p Mar-! Ghost-e ||

```

In order to model this unfolding (of the play) in LINCOS we need eight programs.

The initial state vector is $\parallel \text{Fran-p} \parallel$. Each program “registers” only changes. The first two are

```

DEFINE P1 : Proc → Proc :=
[ * :  $\parallel \text{Fran-p} \parallel$ ] (arb  $\parallel \text{Fran-p} \parallel$ 
 $\parallel \text{Fran-p Ber-e} \parallel$ ).
DEFINE P2 : Proc → Proc :=
[ * :  $\parallel \text{Fran-p Ber-e} \parallel$ ] (arb  $\parallel \text{Fran-p Ber-e} \parallel$ 
 $\parallel \text{Fran-p Ber-p} \parallel$ ).

```

...

And the last program is

```

DEFINE P8 : Proc → Proc :=
[ * :  $\parallel \text{Fran-p Fran-c Ber-p Ber-c Hor-p Hor-c Mar-p}$ 
 $\text{Mar-c} \parallel$ ]
(arb  $\parallel \text{Fran-p Fran-c Ber-p Ber-c Hor-p Hor-c Mar-p}$ 
 $\text{Mar-c} \parallel$ 
 $\parallel \text{Fran-ex Ber-p Hor-p Mar-p Mar-! Ghost-e} \parallel$ ).

```

Note that the following changes are registered with P8.

```

 $\parallel \text{Fran-p Fran-c} \parallel \rightsquigarrow \parallel \text{Fran-ex} \parallel$ ,
soldier FRANSISCO leaves the platform
 $\parallel \text{Ber-c Hor-c} \parallel \rightsquigarrow \parallel \text{Ber-p Hor-p} \parallel$ ,
officers BERNARDO and HORATIO are speechless
 $\parallel \text{Mar-c} \parallel \rightsquigarrow \parallel \text{Mar-!} \parallel$ ,
HAMLET's friend MARCELLUS sees the Ghost!

```

At the same time it is seen that $\parallel \text{Ber-p Hor-p Mar-p} \parallel$ is kept in the new state vector—BERNARDO, MARCELLUS and HORATIO are in place on the platform. This is necessary. Should we have left out one of these, the person would have disappeared, possibly temporarily. That is explicitly the case for FRANSISCO, depending on the interpretation of exit. Complete disappearance would be all right for the GHOST (later on), but not for the officers and the friend of HAMLET!

Conclusion

Processes running in parallel, subjected to interrupts, are modelled in this chapter satisfactorily in terms of the *Lingua Cosmica*. There is no reference to time. Processes carry out elementary steps in a way, either sequentially or in parallel, not related to time intervals. The same applies to interrupts, modelled as changes in state vectors.

References

- A. Ollongren and D. Vakoch *Processes in Lingua Cosmica*, in 'Communication with ExtraTerrestrial Intelligence' (Edited by Douglas A. Vakoch, Suny Press, State University New York), AbSciCon 2010, Texas USA, Session 'Search for Intelligent Life', topic 'Interstellar Message Construction: Can We Make Ourselves Understood?', (2011) 413–418
- A. Ollongren *Processes in Lingua Cosmica* (source Shakespeare's Hamlet), Acta Astronautica 71 (2012), 170–172

Part VI

Symbolic Computation

Introduction

The problem of constructing messages for ETI concerned with mathematical modeling of certain physical phenomena is considered in the chapters of this part. The *Lingua Cosmica* system for interstellar communication has been described before in a number of papers by the present author aimed at specific applications. The present monograph details and places the system in a broad context. It has been mentioned in several places that a central position is occupied by concepts from constructive logic. Therefore an important aspect of using that system in message construction is that, also mentioned several times, guaranteed correct logical reasoning can be achieved. The proposed LINCOS in pure form is, however, not immediately applicable for describing mathematical modeling of aspects of physical reality because of the limited computational power of it. The chapters in this part use a slight, unorthodox modification to the underlying logic, in order to introduce aspects of computational nature, in fact symbolic computation. At the same time the concept of strict typing in constructive logic is modified by introducing a restricted form of nondeterminism in definitions.

The extended form of the *Lingua Cosmica* is called LINCOS⁺. The way these new elements can be put to use is illustrated by showing how the basics of relativistic particle motion can be formulated in logic terms—see Chapters 20 and 21. The concept of proper time appears. Contrasting this, in classical non-relativistic astrodynamics, the proper time is the same as the absolute or universal time—see Chap. 22. A platform for discussing these issues is obtained by formulating the metrics for the time-spaces. These are elegantly incorporated in LINCOS⁺. Correct reasoning is conserved.

Chapter 20

Basics

*Bedyrande vår oskuld sökte vi
att utan formler popularisera
och på det språk de flesta levde i
en blygsam skymt ar klarhet hopsummera.
Verse 4 in Poem 31 of
ANIARA, En revy om människan i tid och rum,*

Harry Martinson, Stockholm 1956

*Assuring our innocence we endeavoured
without formula's to popularize
and, using language most have favoured,
a modest glimpse of clarity to summarise.*

*But this language meant to explain everything
remained unclear to us ourselves, a blindman's game
words which evaded words and played blindness
in the middle of clarity, the soul of the universe.*

*Then we tried to signal as if to savages
and primitive folks of the kind
one reads about from the eon which forms
the temporal lowest level in the day of the spirit.*

*We wrote figures like trees and shrubs
we drew a meandering river
using these in order to compose texts
which they might understand somewhat aided by the pictures.*

*But as we ourselves only found foreign sounds
in that language so far from the land of formulas
we hardly ourselves understood the lessens
by which we wished to stretch out our hand.*

*And the end was that this court of justice
which would liberate us from the verdict of the universe
became differentiated in absurdum
but the bridge between us was equally empty.*

Intention

Pure LINCOS is mainly used for (correct) logical reasoning and it has only limited computational power. In order to be able to include some modest mathematical reasoning in the system we are concerned in the present chapter with expressing the *results* of mathematical operations as addition, multiplication, division, square roots, etc. in the *lingua*. So some restricted form of computation is introduced in the present chapter, in fact by means of a function EVAL, to be used in evaluations. This function, defined below, applied to a mathematical expression is supposed to deliver the evaluated expression of type Set. This is some kind of β -conversion, but distinctly different from the conversion discussed in Appendix D. It is adapted to specific needs as explained below. The *Lingua Cosmica* enriched in this way is called LINCOS*.

Introduction

Mathematical concepts relevant for mathematical modelling of physical phenomena can be assumed to be universal. However, the formalisms as we on Earth use them today in discourses, the notations for the spaces used, the operations over them, reduction mechanisms, etc. have evolved greatly in the course of centuries. They are the result of a long period of developments on the planet in the science of describing physics in mathematical terms. They contain therefore a multitude of ad hoc agreements, notational conventions developed for economy of notation and special symbols with specific interpretations. This means *grosso modo* that it is not feasible to construct interstellar messages concerned with (astro) physical phenomena using exclusively our ordinary mathematical notation.

We have remarked several times before that a message for interstellar communication, written in some unknown symbolism, supposedly meaningful, but exempt from direct clues for its interpretation, presents formidable challenges to receiving parties. To begin with

- How to decide whether the meaning of a message relies on some spoken/written language or on some mathematical system or formal logic?
- If a natural language is behind the message, are simplifications being used, and if so are these of a syntactic and/or a semantic character?
- and then
- If there is no (natural) language behind a given message because the underlying modelling technique is *mathematical*, how to interpret the expressions?

The task of interpretation might also in the latter case greatly be simplified if a message is multilevel, i.e. contains at another level or levels information *about* the message in an abstract system as in the case of our *Lingua Cosmica*. Yet, a message containing mathematical reasoning, annotated by means of abstract terms in terms of some cosmic *lingua*, can only be understood if

1. The receiver finds out and realises what the basis is of the utilised *lingua* (in our case it is constructive logic) and
2. Extralinguistic clues are included to explain that certain basic mathematical constructs are used as well.

It should be an easy matter to inform recipients of a message annotated by terms in LINCOS⁺ how to represent the natural numbers, integers and rational numbers. We could sum up a finite set of natural numbers and we could use examples including arithmetical operations, leading to the infinite set of integers. Statements that every number has a successor 1 larger than it and a predecessor 1 less than it should also be easy to convey. The countable infinite set of rational numbers can also be summed up, starting with the display (using only a finite number of rationals).

CONSTANTS

0, 1, 2, 1/1, 3, 2/1, 1/2, 4, 3/1, 2/2, 1/3, 5, 4/1,
3/2, 2/3, 1/4, 6, 5/1, 4/2, 3/3, 2/4, 1/5 : Set.

There is of course a rule specifying how the next rational is defined, but the finite sequence above contains enough information for “readers” to infer the summing up rule and to conclude that each rational number is of the basic type Set.

A picture showing a rectangular triangle with two vertices of length 1 can be used to explain that we are familiar with $\sqrt{2}$, an irrational number, and in the wake of this explain notational conventions for real numbers. Perhaps we should in addition explain that quite generally algebraic equations characterise real numbers as well. Other geometrical figures may be used to convey the notion of transcendental numbers, e.g. the ratio between the circumference of a circle and its diameter (π).

Pure LINCOS is well suited for (correct) logico-mathematical reasoning. As an example: commutative laws over addition and multiplication are easily expressed as hypotheses. If so is done, the following facts over natural numbers are provable within the system (using here ALL and supposing that the meaning of the operators + and $\times$ is recognised by receivers):

```
FACT comm-add : (ALL a, b : nat)
  (+ a b)  $\rightarrow$  (+ b a).
FACT comm-mult : (ALL a, b : nat)
  ( $\times$  a b)  $\rightarrow$  ( $\times$  b a).
```

LINCOS Enriched

The described *Lingua Cosmica* so far has only limited computational power. Since we must be able to express the *results* of operations as addition, multiplication, division, square roots, etc. over certain collections of objects, we introduce a function EVAL, used in mathematical evaluations. This function, defined below, applied to a mathematical (symbolic) expression delivers the evaluated expression of type Set.

That is effectuated by making use of a definition so that β -conversion, briefly discussed in Appendix, is applicable.

DEFINITION EVAL : math-expr $\rightarrow$ Set := [x : math-expr]
(value-of-expression x).

Here we stipulate that symbolic expressions as they occur in computer algebra are of type math-expr. So we need to enrich our language—as mentioned we call the enriched system LINCOS⁺. The type of (value-of-expression x) is Set and x is supposed to be reduced to basics by means of computer algebra. Using β -conversion (see Appendix D) we have for example

(EVAL 2) = (value-of-expression 2) = 2 : Set
(EVAL (2.1 + 3.7)) = (value-of-expression 5.8) = 5.8
: Set
(EVAL (a.b) / b) = (value-of-expression a) : Set.

Note that until now there is, apart from reduction and evaluation, no computation involved. We proceed with another unconventional step by enriching the original LINCOS here and now even more with concepts from *symbolic computation*. Using infix notation for the arithmetical operators, we allow

(value-of-expression n) = n

for any real number n. Also allowed are reductions as

(value-of-expression (a.b)/b) =
(value-of-expression a) = a

Both these examples are in the realm of symbolic computation.

When discussing physical phenomena we shall evidently need the notions of *units* of length (meter), area (square meter), time (second) and scalar velocity (meter per second). Here we accept a weakness in the discussions due to the fact that we have not introduced the concept of time explicitly. So strictly speaking the interpretation of sec is left open. So the units are introduced in LINCOS⁺ by type declarations:

CONSTANTS m, m², sec, m/sec : Set.

The type of scalar velocity is declared by

CONSTANT velocity : Set $\rightarrow$ m/sec.

Assuming the usual notation for large numbers, specific velocities can be defined, such as the velocity of light:

DEFINITION c : m/sec := (velocity 3.10⁸).

We shall need multiplication in several forms, so in view of the constants above we overload the multiplication sign $\times$, introducing at the same time *non-determinism* in typing:

CONSTANT $\times$: m/sec $\rightarrow$ sec $\rightarrow$ m.
CONSTANT $\times$: m $\rightarrow$ m $\rightarrow$ m².

This looks strange and in conflict with our earlier stipulation: an entity has exactly one type. The situation is not as bad as it looks: if the entity $\times$ is used the type of it is assumed to depend on the context in which it appears. There are two distinct entities, so one might give them different designations to keep LINCOS⁺ pure.

Supposing a and b to be of the right types, we shall not use for multiplication the prefix notation $(\times a b)$, but the customary infix notation $(a.b)$ or $(a \times b)$. Division and the square root as types can be introduced as

CONSTANT $\div$: $m \rightarrow m/\text{sec} \rightarrow \text{sec}$.
 CONSTANT $\sqrt{}$: $m^2 \rightarrow m$. (over area's).

Metric Two-Dimensional Space

When discussing particle motion in a cosmological sense we shall need differentiable metric spaces for which distances are defined. If x_1 and x_2 in the space are both of type meter, $|x_1 - x_2|$ is well defined and we could write Δx for this distance. In case the difference is taken arbitrarily small we replace Δx by the *differential* dx . From the point of view of explaining concepts this is a non-trivial step taken: in fact the realm of calculus is entered. It is of course uncertain whether a step of this kind can be understood by parties without knowledge of elements in calculus. Some help is provided by the observation that we have here a kind of dimensional analysis.

In LINCOS⁺ the differential is introduced by

VARIABLE dx : m .

In the same way for an interval in time measured by an observer in an inertial frame of reference, we introduce the differential dt by

VARIABLE dt : sec .

Then in view of c : m/sec and $\times$: $m/\text{sec} \rightarrow \text{sec} \rightarrow m$, we have

$c.dt$: m , and using $\times$: $m \rightarrow m \rightarrow m^2$, we find $(c.dt).(c.dt)$: m^2 . For the area $dx \cdot dx$ we use

VARIABLE $(dx)^2$: m^2 .

Most of these preparations are used in the next chapter for the discussion of core concepts in the (special and general) theory of relativity. We also need the notion of the line element in two-dimensional space-time with Cartesian co-ordinates x and $c\Box t$. This leads to the concept of the metric of the space. Based on the metric one obtains then a flat, possibly (within itself) *stretched* and/or *curved* space-time. From this also the phenomenon of *time dilatation* can be derived elegantly.

Introductory declarations and definitions are collected here. The list suggests also that we have a kind of dimensional analysis at hand.

CONSTANTS $m, m^2, \text{sec}, m/\text{sec}$: Set.
 CONSTANT velocity : Set $\rightarrow m/\text{sec}$.
 DEFINITION c : $m/\text{sec} := (\text{velocity } 3.10^8)$.

```

VARIABLE dx : m.
VARIABLE ds : m.
VARIABLE (dx)2 : m2.
VARIABLE (ds)2 : m2.
VARIABLE dt:sec.
VARIABLE dt : Set.
(* overloaded *)
VARIABLE (c.dt)2 : m2
VARIABLE dx/dt : m/sec.
VARIABLE ds/c : sec.

(* proper time dτ *)

```

We have here another case of non-deterministic typing: the type of dt is either `sec` or `Set`. Strictly speaking $(c.dt)^2$ need not be declared separately. The same applies to $(dx)^2$ and $(ds)^2$. The object dx/dt must be explained separately to be a differential *quotient*, referring to the mathematical concept of (ordinary) differentiation, the taking of a derivative.

Reference

Harry Martinson *ANIARA A story about man in time and space*, Stockholm (1956), verses 4 – 9 of Poem 31 (translated by the author with the able aid of Gunvor Ollongren)

Chapter 21

Relativistic Particle Motion

Intention

We consider first the non-relativistic motion of a particle (anonymous because properties of it are not described) in a Euclidean, two-dimensional *flat* space–time. It can be formulated in terms of concepts of LINCOS⁺, i.e. the *Lingua Cosmica* enriched with the elements described in the previous chapter. After that the relativistic cases are treated. To begin with the case of the motion of an object in the two-dimensional flat *linear stretched* space–time is discussed. This case belongs to the regime of Special Relativity Theory (SRT). Then we consider a simple case of relativistic particle motion in a non-Euclidian, three-dimensional flat space–time *stretched in two space dimensions*, also formulated in enriched LINCOS⁺. The regime is here that of the General Relativity Theory (GRT).

The concept of time cursorily and informally introduced in Chap. 20, figures in all cases of motion of objects. We should explain to ETI what we mean by clocks and observers, but we refrain from doing so here. In each case discussed, we do concentrate on the behaviour of time measured by a clock of an observer at *rest* with respect to the moving object (which is supposed to dispose of another clock: the *moving* clock).

Non-relativistic Space

Euclidian two-dimensional flat space–time is characterized by the tuple (c, dt, dx) , where c is the velocity of light (electromagnetic radiation), t is the time variable, x is the length variable. The entities dt and dx are differentials over t and x . Gravitating bodies are absent, so motion in this space is always along straight lines, the so-called *world lines*.

For the purpose of mathematically describing particle motion in this two space–time, we need the concepts of *line element* and the Euclidian *metric*. In addition to these we shall need the *proper time* (mathematically the differential $d\tau$, also called

eigen time because this is the time shown by a clock moving with the particle). In terms of LINCOS⁺ we have

```
VARIABLE line-elem : m2 → Set.
DEFINITION proper-time : m2 → Set := [LAMBDA ζ : m2]
(EVAL ( (√ζ) ÷ c ) ) .
```

In flat Euclidean two-dimensional space-time using $c \cdot dt$ and dx as the co-ordinates, the metric follows from defining the line element ds^2 as

$$(c \cdot dt)^2 + dx^2.$$

This is in fact the Pythagorean theorem. In enriched LINCOS we formalize this by

```
DEFINITION EUCL-line-elem-def : m2 → m2 → Set :=
[LAMBDA α, β : m2](line-elem (α + β)) .
```

Note that we write $\alpha + \beta$ in view of the definition of ds^2 . What is left now is the definition of the Euclidian metric as follows:

```
DEFINITION EUCL-metric : m2 → m2 → m2 → Set :=
[LAMBDA α, β, γ : m2](line-elem α) →
(EUCL-line-elem-def β γ) → (proper-time β) .
```

Note: (line-elem α) can be omitted, but is kept for reasons of perspicuity.

Under these definitions

```
(EUCL-metric ds2 (c.dt)2 dx2) :
(line-elem ds2 ) → (line-elem ((c.dt)2 + dx2) →
(proper-time (c.dt)2) .
```

Furthermore $(\text{proper-time } (c \cdot dt)^2) = dt$, because via symbolic computation

```
(EVAL (c.dt)2) ÷ c = dt .
```

Note: $(\text{proper-time } (c \cdot dt)^2) : \text{Set}$ because $dt : \text{Set}$ using the overloaded type of dt .

Mathematically: relative proper-time $d\tau/dt = 1$.

The moving clock synchronizes with the clock of the observer.

The foregoing allows us to state a fact:

```
FACT F1 : (line-elem ds2) → (line-elem ((c.dt)2 + dx2)
→ (proper-time (c.dt)2) → dt .
```

Because $(\text{proper-time } (c \cdot dt)^2) = dt$, the proof is

```
F1 = [LAMBDA p : (line-elem ds2); q : (line-elem ((c.
dt)2 + dx2); r : (proper-time (c.dt)2)]r .
```

Special Relativity Theory in LINCOS⁺

For relativistic two space–time with rectangular co-ordinates $c \cdot dt$ and dx , the metric is based on the following definition of the line element

$$ds^2 = (c \cdot dt)^2 - dx^2.$$

This choice is motivated by the physical law stating that the velocity of light emitted by a moving object is equal to the universal constant c (velocity of light), independent of the emitter's motion. The *proper* time $d\tau$ is in general not equal to dt in this case. There is *linear stretching* in the one-dimensional space. The LINCOS⁺ formalization leading the metric for this case is as follows (compare with EUCL-line-elem-def):

```
VARIABLE line-elem : m2 → Set.
DEFINITION proper-time : m2 → Set := [LAMBDA ζ : m2]
(EVAL ( (√ζ) ÷ c )) .
DEFINITION SRT-line-elem-def : m2 → m2 → Set :=
[LAMBDA α, β : m2]
(line-elem (α - β)) .
```

Note that we write $\alpha - \beta$ in view of the definition of ds^2 . This leads to the (special relativistic) metric

```
DEFINITION SRT-metric : m2 → m2 → m2 → Set :=
[LAMBDA α, β, γ : m2] (line-elem α) → (SRT-line-
elem-def β γ) → (proper-time (β - γ)) .
```

The proper time depends on the trajectory in two space–time. Because we have flat space (in the absence of gravitation generated by massive objects) a world line is a straight line. We consider three cases.

1. Let $|dx/dt|=c$. Trajectory of electromagnetic radiation. Then

```
(SRT-metric ds2 (c.dt)2 (c.dt)2 ) :
(line-elem ds2) → (line-elem ((c.dt)2 - (c.dt)2 ))
→ (proper-time ((c.dt)2 - (c.dt)2 )) .
```

Via symbolic computation we have

```
(proper-time (c.dt)2 - (c.dt)2) = 0 : Set, because
(EVAL (√((c.dt)2 - (c.dt)2)) ÷ c) = 0 is of type Set.
```

This allows us to state a fact:

```
FACT F2 : (line_elem ds2) → (line_elem ((c.dt)2 -
(c.dt)2 )) → (proper-time ((c.dt)2 - (c.dt)2 )) → 0.
```

The proof is similar to the proof of fact F1.

Mathematically: $ds^2 = (c \cdot dt)^2 - dx^2 = 0$, because $|dx| = c \cdot |dt|$ and relative proper (eigen) time $d\tau/dt = ds/(c \cdot dt) = 0$.

The moving clock stands still.

2. Let $dx=0$. The object is at rest with respect to the observer. Then

$$\begin{aligned} (\text{SRT-metric } ds^2 \text{ } (c \cdot dt)^2 \text{ } 0) &: (\text{line-elem } ds^2) \rightarrow \\ &(\text{SRT-line-elem-def } (c \cdot dt)^2 \text{ } 0) \rightarrow \\ &(\text{proper-time } (c \cdot dt)^2). \end{aligned}$$

This case is similar to motion in Euclidean space. *Mathematically:* $ds^2 = (c \cdot dt)^2$, so $|ds| = |c \cdot dt|$, $|ds/c| = |dt|$ and relative eigen time $|d\tau/dt| = |ds/(c \cdot dt)| = 1$.

The moving clock synchronizes with the inertial clock.

3. Let $|dx/dt| < c$. This is the most general case, that of subluminal velocity of an object. Then

$$\begin{aligned} (\text{SRT-metric } ds^2 \text{ } (c \cdot dt)^2 \text{ } dx^2) &: (\text{line-elem } ds^2) \rightarrow \\ &(\text{SRT-line-elem-def } (c \cdot dt)^2 \text{ } dx^2) \rightarrow \\ &(\text{proper-time } ((c \cdot dt)^2 - dx^2)). \end{aligned}$$

Mathematically:

$$ds^2 = (c \cdot dt)^2 - dx^2 < (c \cdot dt)^2,$$

so $(ds/c)^2 < dt^2$ and as a result $|d\tau| < |dt|$, relative proper time < 1 .

The moving clock slows down.

This is the phenomenon of *time dilatation*.

General Relativity (GRT) in LINCOS⁺

We consider next a non-Euclidean three space–time with at the origin a mass point M . Let the time differential co-ordinate be $c \cdot dt$ as before, and let the two polar (also rectangular) differential co-ordinates be dr for radius and $r \cdot d\theta$ perpendicular to it.

For $M=0$, the metric is then defined by the line element:

$$ds^2 = (c \cdot dt)^2 - dr^2 - (r \cdot d\theta)^2.$$

The metric for the special case SRT is recovered for $d\theta=0$, and $dr^2 = dx^2$.

For $M>0$ the metric is the so-called Schwarzschild metric:

$$ds^2 = (c \cdot dt)^2 (1 - 2G \cdot M / (r \cdot c^2)) - (dr)^2 / (1 - 2GM / (r \cdot c^2)) - (r \cdot d\theta)^2.$$

Here G is the *gravitational constant*. With this choice of the metric the velocity of light is again set to c , the velocity of electromagnetic radiation, a universal

constant. For large values of r , we have approximately $ds^2 = (c \cdot dt)^2 - dr^2 - (r \cdot d\theta)^2$, the same as for $M=0$.

Note that the term $1 - 2GM/(r \cdot c^2)$ is neutral with respect to dimensions.

Next we need the metric in GRT.

```
VARIABLE line-elem : m2 → Set.
DEFINITION proper-time : m2 → Set := [LAMBDA ζ : m2]
(EVAL ( (√ζ) ÷ c )) .
DEFINITION GRT-line-elem-def : m2 → m2 → m2 → Set :=
[LAMBDA α, β, γ : m2] (line-elem (α - β - γ)) .
```

Here we have the line element in GRT in three space-time; we write $\alpha - \beta - \gamma$ in view of the definition of ds^2 . We must keep in mind

$$\begin{aligned}\alpha &= (c \cdot dt)^2 \cdot (1 - 2GM / (r \cdot c^2)), \\ \beta &= (dr)^2 / (1 - 2GM / (r \cdot c^2)), \\ \gamma &= (r \cdot d\theta)^2.\end{aligned}$$

The required metric is

```
DEFINITION GRT-metric : m2 → m2 → m2 → m2 → Set :=
[LAMBDA α, β, γ, δ : m2] (line-elem α) → (GRT-line-
elem-def β, γ, δ) → (proper-time β - γ - δ) .
```

The three cases discussed for the special case SRT (electromagnetic radiation, object at rest and subluminal velocities) are recovered in GRT for $d\theta=0$, and $dr^2=dx^2$. We have then a non-Euclidean two space-time and motion is in the direction of the mass point M . The space stretching of the two space-time is that of the SRT. Consider now the case $d\theta$ unequal to 0, the case of a non-Euclidian three space-time

$$ds^2 = \Phi \cdot (c \cdot dt)^2 - \Phi \cdot 1 \cdot (dr)^2 - (r \cdot d\theta)^2.$$

as before, using the shorthand notation $\Phi = (1 - 2GM/(r \cdot c^2))$, adapted from the Russian фактор (factor). In addition, using the argument $\alpha - \beta - \gamma$ for line-elem in view of the definition of ds^2 we have

```
VARIABLE line-elem : m2 → Set.
DEFINITION proper-time : m2 → Set := [LAMBDA ζ : m2]
(EVAL ( (√ζ) ÷ c )) .
DEFINITION GRT-line-elem-def : m2 → m2 → m2 → Set :=
[LAMBDA α, β, γ : m2] (line-elem (α - β - γ)) .
```

DEFINITION `GRT-metric` : $m^2 \rightarrow m^2 \rightarrow m^2 \rightarrow m^2 \rightarrow \text{Set} :=$
 $[\text{LAMBDA } \alpha, \beta, \gamma, \delta : m^2](\text{line-elem } \alpha) \rightarrow (\text{GRT-line-}$
 $\text{elem-def } \beta, \gamma, \delta) \rightarrow (\text{proper-time } \beta - \gamma - \delta).$

So in general

$$(\text{GRT-metric } ds^2 \phi.(c.dt)^2 \phi^{-1}.(dr)^2 (r.d\theta)^2) : \\
(\text{line-elem } ds^2) \rightarrow (\text{GRT-line-elem-def } \phi.(c.dt)^2 \\
\phi^{-1}.(dr)^2 (r.d\theta)^2)) \rightarrow \\
(\text{proper-time } (\phi.(c.dt)^2 - \phi^{-1}.(dr)^2 - (r.d\theta)^2)).$$

Now the proper time depends on the central mass M and on θ . For $M=0$ we have $\Phi=1$, and then the proper time is $(\text{EVAL } \sqrt{[(c.dt)^2 - (dr)^2 - (r.d\theta)^2]}/c)$.

We have here a non-Euclidean three space-time and as before the value of the *proper* time depends on the trajectory of a moving particle. Not only the motion of the particle is curved in two space (two dimensions), the two space itself is stretched and curved in itself.

Conclusion

LINCOS based on strict typing as used in constructive logic has been shown to be a useful tool for formulating annotations of large-size messages for ETI. The logic contents can be summarized in rather small sections. For the case that mathematical formulations describing physical phenomena are included in messages, the two chapters in the present PART VI demonstrate that a modified kind of typing and reduction is useful. Two modifications are considered: the introduction of non-deterministic typing and the use of symbolic computing. The first consists of overloading type definitions and the second is realized by allowing elements of symbolic computing in LINCOS. By means of examples the use of these devices is explicated. Note that the formulation of essentially mathematical reasoning in LINCOS is *not* supposed to replace the mathematics. These can be kept, even *in situ*. The supplementary logic terms are again meant for annotation (Ollongren 2004).

A special advantage of this way of considering the problem construction of large-size messages for ETI is that basically the same kind of logic is used for annotations—whether one is concerned with logic or mathematical contents. On the other hand if the receiving party knows the particular type of mathematics we use for some application, then the kind of annotations proposed here may serve to clarify the conventions of LINCOS. This then is a completely different kind of clarification method than that has been considered in earlier chapters: self-interpretation (Chap. 13) and using music (Chap. 15) for that purpose.

Reference

- A. Ollongren *Symbolic Computation in LINCOS*, Accepted poster for the 8th International Conference on Bioastronomy, Reykjavik, July 12–16 (2004), Section ‘Intelligent Life’ beyond the Solar System’.

Chapter 22

Two-Body Motion

Intention

The three space–time discussed in Chap. 21 is, apart from the central mass, an empty space as far as matter is concerned. Embedded in the two-dimensional subspace is a force field, in the form of loci of equal gravitational “pull”, concentric with respect to the mass M . But this field is itself rather “untangible” and there is no trace of gravitational effects as long as interactions with another body or several other bodies are not taken into account. In view of the structure of the solar system of the Sun and planets orbiting around it, the three space–time is, though mathematically speaking certainly interesting, in many respects unnecessarily complicated. The present chapter introduces a simple and useful realistic model: in two-dimensional Euclidian space besides the central mass M , there is a second non-zero mass M , located and moving somewhere in the space. We consider here only non-relativistic motion. This is a basic situation in celestial mechanics, the science of mathematically describing the motions of bodies in the solar system (Danby 1962).

Co-ordinates

In a two-dimensional space with a central mass M and a secondary mass m we have two mutually distinct fields of force superimposed on the plane. One cannot assume both of them to be static, because any initial configuration where M and m occupy distinct positions and have zero velocities is unstable. So one is interested in the dynamics of the system, where the force fields admit an equilibrium condition. Directed gravitational attraction occurs between M and m , but also between m and M . We shall discuss in the present chapter the non-relativistic motion of the two bodies under the influence of the mutual gravitational attraction.

The Euclidian metric in two space using polar co-ordinates r and $r \cdot d\theta$ is

$$ds^2 = dr^2 + (r \cdot d\theta)^2.$$

That is, the Pythagorean theorem in the plane applies. Note that because we treat a non-relativistic case, time is absent in the metric. Because we are interested in motion, time is there in the background as a universal, absolute concept—and the value of it is independent of the state of motion of a time-measuring device, generally a clock. Therefore the concept of proper time, distinct from absolute time, is absent.

The situation of two objects in orbits in the Euclidian plane, mentioned above, is detailed and simplified for ease of analysis as follows:

- Both M and m are point-masses.
- m is much smaller in mass than M , so $m \ll M$.
- M is stationary in the plane (this approximation is customary in elementary astrodynamics).
- m moves in a circular locus (radius r) in the plane under the gravitational attraction of M .

The last supposition is not completely realistic and is only used as an ideal case (the case of circular motion in astrodynamics, not exactly fulfilled in the case of the solar system).

Orbital Elements

Differentiating the Euclidian metric we find for circular motion ($dr=0$)

$$ds = r \cdot d\theta, \quad ds / dt = r \cdot d\theta / dt, \quad d\theta / dt = v / r,$$

with $v=ds/dt$, the velocity of m .

The centrifugal force experienced by m due to motion in the circular orbit is $m \cdot v^2 / r$. The force exerted by M on m is $G \cdot m \cdot M / r^2$. G is the gravitational constant. A stable situation where the forces cancel one another is achieved for

$$m \cdot v^2 / r = G \cdot m \cdot M / r^2 \text{ from which we find } v^2 \cdot r = G \cdot M.$$

The length of the circular orbit is $2\pi \cdot r$, so the period of the motion is

$$P = 2\pi \cdot r / v = 2\pi \cdot r^{3/2} / (G \times M)^{1/2}.$$

We have here a simplified form of Kepler's famous *third law* ([Kepler](#)) for planets moving around their Sun. The general law states that if P is the (sidereal) period of the motion of a planet, and r the mean distance from the Sun, the square of P is proportional to the third power of r . An example: The mean distance between Jupiter and the Sun is about 5.2 times the mean distance between the Earth and the Sun. Therefore the (sidereal) period of Jupiter is about $(5.2)^{3/2}$ years, i.e. approximately 11.8 years.

Euclidean Metric

The metric in two-dimensional space can be defined as $ds^2 = dr^2 + (r \cdot d\theta)^2$ and can be formalised in LINCOS⁺ like this:

```
VARIABLE line-elem : m2 → Set
DEFINITION EUCL-line-elem-def : m2 → m2 → Set :=
[LAMBDA α, β : m2](line-elem (α + β))
```

using again the meter m as the unit of length. The metric itself is defined by

```
DEFINITION EUCL-metric : m2 → m2 → Set :=
[LAMBDA α, β : m2](line-elem α) → (EUCL-line-
elem-def α β).
```

Since time is absolute, the concept of proper time, distinct from absolute time, is indeed absent.

The Euclidian metric above, together with absolute time and the laws of gravitation, constitutes the base of all theories of motion in the planetary system, be it motion of the planets, small planets, asteroids, satellites of planets or artificial objects in orbit.

Conclusion

If ExtraTerrestrial Intelligence exists it is likely to be situated on a planet in a planetary system. So in communication with ETI, it is not unreasonable to include in messages from Earth some kind of description of our situation in the Solar System. A neutral way of doing so would be by including a description of the dynamical properties of a planetary system: the fact that a planet moves in a plane and that the planets move more or less in the same plane. Other solar systems, if consisting of one central star and one or more planets orbiting around it, will *per force* have to be subjected to the laws of mechanics as we know them governing the motions. The three laws of Kepler must be known to alien intelligent inhabitants of a distant planet in such a planetary system. So why not inform—in the form of LINCOS⁺ descriptions—the “others” about our system of Sun and Planets.

References

There are many introductions to Celestial Mechanics (often called Astrodynamics), one of them being the excellent, older elementary book *Fundamentals of Celestial Mechanics*, published by J.M.A. Danby, The Macmillan Company (1962). The author's handout lecture notes 'Hemelmechanica', Leiden University (1999), were not published.

Kepler's famous three laws of motion are concerned with the motions of the planets around the Sun. The first *two* laws are: (1) the orbit of a planet is an ellipse in a plane, with the Sun at one of the *foci*, (2) a planet moves such that the moving straight line connecting it to the Sun, covers in an interval Δt of the universal time t , an area proportional (via a constant of motion), to Δt , independent of its position in orbit.

Part VII (Un)Certainty

Introduction

The two chapters of this part are of an unusual character, as both certain existence and uncertainty are considered. In a way we return to pure LINCOS as described in Parts I–V. In terms of typed constructive logic (the basis of the *Lingua Cosmica* discussed in this book) a given type is said to be justified if an entity can be shown to be of that type, either by definition, by hypothesis or by construction. In Chap. 2 of Part I functional existence in a general sense and depending on a predicate is discussed. Alternatively one can focus on the entity itself and consider it to *exist* if it justifies a type. We have in that case a predicate-free existence, a kind of absolute certainty. As always a certainty achieved in some way is based on what is called an environment (or stage) containing relevant certain information augmented with constructive, often extensive argumentation.

In the present part we focus first on existence as a certainty and describe some certainties in the world of male and female humans. With examples we show that in order to restrict relations seemingly logical in that world, broader knowledge of the world is crucial. It is indeed well known that without such knowledge logically correct but unrealistic conclusions can easily be drawn, even with a sound environment in the background.

A sound environment is free of contradictions, but might not be “rich” enough for satisfactory interpretation of formulas. In order to illustrate aspects of this situation a somewhat unusual intelligent being appears on the scene: the *uncertain alien*. The uncertain alien is the one who has operational knowledge of the meaning of truth but is unaware of falsity. How to deal with uncertainty?

Chapter 23

Certain Existence

Intention

Designers of sizable interstellar messages might easily be faced with the problem of how to deal with existential aspects as they occur in our “knowledge of the world”. Surely topics to be described should exist, but in what sense? Scenes described in poetry “exist” in a completely different way than descriptions of happenings in a historical survey. Whatever formal system is utilized to express aspects of human societies, it should be able to handle existence and representations of it in a logical sense. *De facto* the concept is explicitly predicatively used in earlier chapters describing various aspects of *Lingua Cosmica*. One purpose of the present chapter is to demonstrate how existence, regarded as a kind of impredicative certainty, can be embedded in *Lingua Cosmica*. After that we return to the predicative case discussing existence maps for the so-called *all* case and the so-called *some* case. An example using these is provided.

Introduction

From previous chapters as detailed in Parts I–V of this book, it should be clear that the notion of logical *existence* is available in LINCOS—it is introduced separately in the discourse of Chap. 2. This kind can be called *predicative* existence since a predicate is involved. In the present chapter the notion is provided with an alternative aspect: we consider existence as a kind of certainty. This goal is achieved by giving a slightly different interpretation to the generic type expression $a : A$, where the type A is in the environment. Instead of stating that a (itself a type) justifies A (or a lives in A) we say in this chapter that a of type A *exists*. In this case we have *impredicative* existence as no predicate is involved. In this view it is known not only that A is justified but also for what reason. A itself also exists, because it is in the environment. Like this a form of certainty is available and expressed. *Note*: The present chapter is a successor to paper (Ollongren and Vakoch 2011).

Propositions Exist

Types and type theory occupy central positions in LINCOS, and we have seen that there is a hierarchy (a partial ordering) between the basic types Prop, Set and Type because $\text{Prop} : \text{Type}$ and $\text{Set} : \text{Type}$, Prop and Set can be said to exist, but Type falls outside of this conceptualization. The type Set is not a representation of mathematical sets. The latter can be represented as types, but types are more general than sets. Mathematics concepts as set membership, subsets of sets and empty set, as well as operations over sets, are expressible in type theory, but not all types can be represented as sets.

In the present chapter we again use only one basic type, the one called Prop.

Let X , considered to be a variable, be *of type* Prop, writing

VARIABLE $X : \text{Prop}$.

stating that X exists. We have seen that in constructive logic one says that Prop is *justified* by X . We can also say that Prop *is the case*. Prop can be justified by many mutually distinct logical expressions. The entity X of type Prop is considered to be a *proposition*, not in the classical sense of being true or false, but in the sense that it can occur in expressions built of propositions of type Prop using logical connectives. One can also state that X is an existing term of type Prop.

We have seen that a strict rule stipulates that an expression justifies one and only one type, i.e. it has exactly one type. Note that a type cannot justify itself (except Type!). Prop exists. In addition to this, if an expression exists, it has a unique type. The implication $X \rightarrow \text{Prop}$ has type Prop. Note, however, that the token $\rightarrow$ is overloaded and can be used in a declaration of a *map*, for instance like this:

VARIABLE $P : X \rightarrow \text{Prop}$.

In this the type of $X \rightarrow \text{Prop}$ must be in the hierarchy of basic types, “superior” to Prop, so the type must be Type. Note that P of type $X \rightarrow \text{Prop}$ exists, but evidently cannot be interpreted as a proposition. We have seen that P is in fact a *predicate* over X . If $x : X$ exists, then P can be *applied* to x , and we have seen $(P\ x) : \text{Prop}$. Thus both X and $(P\ x)$ exist in Prop; each of them is a *bona fide* proposition.

All and Lambda Binding

Assume the declarations of X and P as given above, and their existence is certain. Then for *all* $x : X$ we have $(P\ x) : \text{Prop}$. This is expressed by $(\text{ALL } x : X)(P\ x) : \text{Prop}$, or by writing simply $(x : X)(P\ x) : \text{Prop}$, as we have seen before. So prefixing $(x : X)$ to $(P\ x)$ does *not* change the type. Now assume the declaration

VARIABLE $y : X$.

Then $(P y)$ exists because $(P y) : \text{Prop}$. However, is $(P y)$ itself justified? Remarkably it is generally not so. In order to show this consider now the evident relation:

FACT $f.23.1 : (x : X) (P x) \rightarrow (P y) .$

In order to verify this fact we need lambda binding. In $[H : (x : X)(P x)]$ the variable H is locally given the type (lambda bound to) $(x : X)(P x)$. In that case $(H y) : (P y)$.

So $[H : (x : X)(P x)] (H y) : (x : X)(P x) \rightarrow (P y)$. From which the conclusion is

$f.23.1 = [H : (x : X) (P x)] (H y) .$

Applying $f.23.1$ to y we find the identity map

$(f.23.1 y) : (P y) \rightarrow (P y) .$

All and Some in Existence Maps

We shall use as before the variable $X : \text{Prop}$ and the predicate $P : X \rightarrow \text{Prop}$, not declared but as parameters in definitions. In addition also the predicate $S : X \rightarrow \text{Prop}$ is to be used (in view of the Aristotelian Theatre, Chap. 7). The predicates are not necessarily connected to Aristotelian subjects and predications. For chosen P and S we shall be interested in the following cases:

- for *all* $x : X$, $(S x) \rightarrow (P x)$ is justified.
- for *some* $x : X$, $(S x) \rightarrow (P x)$ is justified.

As explained $(x : X)(P x) : \text{Prop}$ expresses that for *all* $x : X$ we have $(P x) : \text{Prop}$. We need now to consider in addition to this the case that only for *some* $x : X$, $(P x) : \text{Prop}$ is the case. So let $z : X$ be a particular *singular* to be used to model “some x of type X ”. In order to treat the two cases we shall need two existence maps, defined inductively.

For the first case we use

```
INDUCTIVE existence-all [X : Prop; S, P : X → Prop] :
Prop :=
ex-intro-all : (x : X) ((S x) → (P x)) → (existence-
all X S P).
```

Here the *induction hypothesis* $(x : X) ((S x) \rightarrow (P x))$ expresses the condition for $(\text{existence-all } X \text{ } S \text{ } P)$ to be the case. We have

```
ex-intro-all : (X : Prop) (S : X → Prop) (P : X →
Prop)
(x : X) ((S x) → (P x)) → (existence-all X S P).
```

For the second case we need (remembering $z : X$)

```

INDUCTIVE existence-some [X : Prop; S, P : X → Prop] :
Prop :=
ex-intro-some: ((S z) → (P z)) → (existence-some X S P).

```

The *induction hypothesis* is now $((S z) \rightarrow (P z))$. We have then for this case

```

ex-intro-some : (X : Prop) (S : X → Prop) (P : X →
Prop)
((S z) → (P z)) → (existence-some X S P).

```

Procreation

Let it be supposed that in some context humans, males and females, are mentioned.

All males and all females are humans. A couple consists of a male and a female and *some* of them are able to give birth to humans, i.e. offspring. Not all couples are procreative. Knowledge of the world informs that only some men and some women are procreative—for instance, they may not be too young and not too old. Taking into account restrictions imposed by our world is in general a difficult matter in any system for interstellar communication, in particular when the concept of time is absent in the system. Even in the case of processes discussed in Part V of the present treaty, time is not directly available. We introduce in this chapter some simple criteria and use subclasses of men and women as well as other distinguishing qualifications for procreativity of human couples.

We need to note here that the example of procreativity provided is not meant to discuss the subtlety of distinguishing between something that *can* be done and something that *might* be done. The ability of doing something is a necessary prerequisite for actually doing so and is by way of this example modelled in LINCOS. But the case described, even though important in (human) societies, is certainly not elaborated in some detail mainly for incorporation in a message for ETI.

Let the universe of discourse be the (set of) humans, type H, where $H : \text{Prop}$. Now (virile) males and (procreative) females together with (procreative) couples will have to be typified.

First case: All procreative couples that can produce offspring might do so.

```

Male : H. Virile-male : Male → Prop.
Female : H. Procreative-female : Female → Prop.
Procreative-Couple : Prop.
CONSTANT can-produce-offspring: Prop.
CONSTANT might-produce-offspring : Prop.
DEFINE X := Procreative-Couple : Prop.
DEFINE S :=
[h : Procreative-Couple] can-produce-offspring :
Procreative-Couple → Prop.

```

```

DEFINE P :=
[h : Procreative-Couple] might-produce-offspring :
Procreative-Couple → Prop.

```

Note: If $x : \text{Procreative-Couple}$ then $(S\ x) : \text{Prop}$ and $(P\ x) : \text{Prop}$.

```

ex-intro-all : (X : Prop) (S : X → Prop) (P : X → Prop)
(x : X) ((S x) → (P x)) →
(existence-all X S P).

```

The main part of the *induction hypothesis* of existence-all is $(x : X) ((S\ x) \rightarrow (P\ x))$, and expresses the necessary condition for all procreative couples capable of producing offspring to actually do so. Take X , S and P as shown above, and consider the fact

```

FACT f.23.2 :
(x : X) ((S x) → (P x)) → (existence-all X S P).

```

Verification. Let $x : \text{Procreative-Couple}$. NB. $X = \text{Procreative-Couple}$.

```

f.23.2 = [h : (x : X) ((S x) → (P x))] (ex-intro-all X
S P h) because

```

```

[h : (x : X) ((S x) → (P x))] (ex-intro-all X S P h) :
(x : X) ((S x) → (P x)) → (existence-all X S P).

```

Second case: Some procreative couples that can produce offspring might do so. We use now

```

Male : H. Virile-male : Male → Prop.
Female : H. Procreative-female : Female → Prop.
Procreative-Couple-specific : Procreative-female →
Virile-male → Prop.
m : Male is some male, (Virile-male m) : Prop.
f : Female is some female, (Procreative-female f) :
Prop.
(Procreative-Couple-specific f m) : Prop.
CONSTANT can-produce-offspring: Prop.
CONSTANT might-produce-offspring : Prop.
DEFINE X := (Procreative-Couple-specific f m) : Prop.
DEFINE S := [h : X] can-produce-offspring : X → Prop.
DEFINE P := [h : X] might-produce-offspring : X → Prop.

```

Note: Using $z : (\text{Procreative-Couple-specific } f\ m)$, we have $(S\ z) : \text{Prop}$ and $(P\ z) : \text{Prop}$.

Take X , S and P as shown above. We have then for this case

```

ex-intro-some : (X : Prop) (S : X → Prop) (P : X → Prop)
((S z) → (P z)) →
(existence-some X S P).

```

The *induction hypothesis* of existence is $((S\ z) \rightarrow (P\ z))$, expressing the condition that the specific procreative couple (f, m) , capable of producing offspring might do so. Consider the fact

FACT f.23.3 :
 $((S\ z) \rightarrow (P\ z)) \rightarrow (\text{existence-some } X\ S\ P).$

Verification

f.23.3 = $[h : (S\ z) \rightarrow (P\ z)]\ (\text{ex-intro-some } X\ S\ P\ h)\ \text{because}$

$[h : (S\ z) \rightarrow (P\ z)]\ (\text{ex-intro-some } X\ S\ P\ h) : ((S\ z) \rightarrow (P\ z)) \rightarrow (\text{existence-some } X\ S\ P).$

Conclusion

A message designed for transmission to ETI uses a database of hypotheses and facts, i.e. environmental information (see also Wittgenstein's Theatre, in Chap. 8). Whenever such a message is constructed and formulated in terms of *Lingua Cosmica*, care must be taken that the database of hypotheses and facts is *consistent* (sometimes called *sound*)—we have mentioned this before. The present chapter shows using examples of existence, that in addition to this, the environment should be, besides consistent, also as complete as possible—in view of “what to communicate in LINCOS”. Fortunately an environment can be enlarged by adding definitions, facts and hypotheses to it. In the LINCOS system deleting items from the environment (based on some sort of requirements) is not possible. The examples discussed in the present chapter (and elsewhere) show that the database must be constructed with great care, enabling verifications of useful facts.

In addition to these remarks, note that especially the examples given in this chapter indicate that many, even simple existential relations in human society are not easily expressed in the framework of LINCOS.

Reference

- A. Ollongren and D. Vakoch *Logical existence expressed in Lingua Cosmica*, paper IAC-11 A4.1.6, International Astronautical Congress (2011), submitted for publication in Acta Astronautica

Chapter 24

The Uncertain Alien

Intention

An unusual platform in the field of interstellar communication with unknown extra-terrestrial alien cultures is introduced. It is concerned with the concept of *uncertainty* and ways to deal with it in *Lingua Cosmica*. We use in the present chapter the basic supposition that ETI cultures are familiar with the concept of truth, but are not necessarily aware of falsity. What happens if uncertainty prevails?

Background

Research in the field of interstellar communication with alien cultures is evidently confronted with many uncertainties. In the present chapter we shall be concerned with a specific concept of uncertainty and use an unconventional platform for the discussion. The situation could be paraphrased by “We know what we know and what we do not know, but it is uncertain what the aliens know and do not know”. This point of view may have prime relevance in communication with unknown extraterrestrial intelligent cultures. This is a good reason for giving attention in the present chapter to ways and means for dealing with uncertainty in *Lingua Cosmica*.

We use in this chapter the basic supposition that ETI cultures are familiar with the concept of truth, also in an operational way, but are unfamiliar with falsity. *Truth* in some sense can be assumed to be a universal notion, but the concept may have different kinds of strength, varying from “it is strictly so” to “it is reasonably so”. In any case the assumption would seem hard to verify and impossible to falsify. *Falsity*, as known and used by Earthlings in the familiar propositional and predicate logics, need not necessarily be universal. One can very well imagine that advanced technological societies might exist in the Universe, where instead of the notion of falsity the concept of *uncertainty* is used, operationally not equivalent to our concept of falsity. In other words uncertainty is weaker than falsity. One can imagine for

instance that a proposition might be true at first sight in the view of some alien society, but is kept uncertain as long as the truth of it is not firmly established—or cannot be disproved. We consider in this chapter how uncertainty can be treated in the *Lingua Cosmica* explained in this treatise.

Peirce's Law Uncertain

A central notion in LINCOS is the *type expression*. A type expression E can be introduced by giving it the so-called *resident* (or *inhabitant*), for instance c , by the following statement $\text{CONSTANT } c : E$. In this case one states (declares) that c is of type E . Like this the relation $c : E$ is entered in the *environment*, the database of logic expressions. A type expression might, however, be given a resident in the form of a *fact* G by stating $\text{FACT } G : E$. In the latter case a prerequisite strict condition is that the fact is verifiable. If $c : E$ can be constructed and so becomes available, we find $G = c$, a trivial verification. More generally verification of G in a formal sense is achieved by using the constructive machinery available in LINCOS for finding an expression $e : E$, in such a way that $G = e$. See in Chap. 8, Wittgenstein's Theatre, how such constructions can be effectuated.

Suppose that $\text{FACT } G : E$ is stated but fact G cannot be verified, i.e. the mentioned machinery for producing correct expressions is unable to produce an expression e of the required type E . In that case there is no compelling reason to state that “ G is false” (*Tertium non Datur* is not applicable) and G can be considered to be *uncertain*.

An example of this situation is the case of the famous Peirce's law, a tautology in classical logic. Let A and B be propositions declared in LINCOS by $\text{CONSTANTS } A, B : \text{Prop}$. Pierce's law states in terms of classical propositional logic that for any combination of the values true and false for A and B , the implication $((A \rightarrow B) \rightarrow A) \rightarrow A$ is true. Writing T for *truth* and F for *falsity* (in this chapter only!) and remembering that $A \rightarrow B$ yields T unless $A = T$ and $B = F$, because $T \rightarrow F$ yields F , we find the verification of the law in ordinary propositional logic as follows.

We let A and B go through sequences of T and F in parallel such that all possible combinations are covered, and obtain the following:

Let A be the sequence	$T \ F \ T \ F$
Let B be the sequence	$T \ T \ F \ F$
Then $A \rightarrow B$ is	$T \ T \ F \ T$
and $(A \rightarrow B) \rightarrow A$ is	$T \ F \ T \ F$
so that $((A \rightarrow B) \rightarrow A) \rightarrow A$ is	$T \ T \ T \ T$, the
	required proof.

Peirce's law, written in LINCOS as an implication in terms of constructive logic, is

$\text{FACT Peirce} : ((A \rightarrow B) \rightarrow A) \rightarrow A.$

Attention: This fact (a lemma) *cannot* be verified by the constructive proof machinery. For verification of any implication the premise of the expression (possibly extended with information in the environment) belongs to the available means. The premise of the above implication is $H : (A \rightarrow B) \rightarrow A$. There is no reason for assuming $A \rightarrow B$ to be the case. Using the expression for H only it is *not* possible to construct a resident of the conclusion A . This is easily checked using the machinery described in Chap. 8, Wittgenstein's Theatre.

A modification of Peirce's law is $(A \rightarrow (A \rightarrow B)) \rightarrow A$, not itself a law. We find, shortening the notation: $(A \rightarrow B)$ is T T F T, so $A \rightarrow (A \rightarrow B)$ is T T F T, and $(A \rightarrow (A \rightarrow B)) \rightarrow A$ is T F T F in propositional logic. Note that a resident of $(A \rightarrow (A \rightarrow B)) \rightarrow A$ cannot be constructed in terms of LINCOS. This is because using the premise $H : A \rightarrow (A \rightarrow B)$, one cannot justify A . Therefore the above modification of Peirce's law also cannot be justified.

Other modifications of Peirce are the following:

$((A \rightarrow B) \rightarrow A) \rightarrow B$ is T T F T and $(A \rightarrow (A \rightarrow B)) \rightarrow B$ is T T T F

in propositional logic. None of these expressions can constructively be verified as facts.

Conclusion: Peirce's law and the modifications considered could be regarded as *uncertainties* in terms of LINCOS. However, because Pierce's law holds for all A and B in classical propositional calculus, we are allowed to introduce it by the following hypothesis:

HYPOTHESIS Peirce : $((A \rightarrow B) \rightarrow A) \rightarrow A$.

In this manner the entity Peirce has changed from being uncertain to a (certain) constant of a given type. Using the hypothesis a resident of A can be constructed as follows:

VARIABLE $H : (A \rightarrow B) \rightarrow A$.
(Peirce H) : A .

Note that H can also be entered implicitly:

$[H : (A \rightarrow B) \rightarrow A] (\text{Peirce } H) : ((A \rightarrow B) \rightarrow A) \rightarrow A$.

Indeed, if $(A \rightarrow B) \rightarrow A$ is the case, then A is the case. Note that the formal lambda binding of H expressed by $H : (A \rightarrow B) \rightarrow A$ does *not* mean that $(A \rightarrow B)$ is the case.

If $x : A \rightarrow B$ then $(H x) : A$.

Uncertainty Risks

We mentioned in the foregoing two modifications of Peirce's law, implying B .

Let us for the moment ignore Pierce's law in the form of a hypothesis. In view of the premises $H1 : (A \rightarrow B) \rightarrow A$ and $H2 : A \rightarrow (A \rightarrow B)$ it is no surprise that the facts

FACT Pierce-m1 : $((A \rightarrow B) \rightarrow A) \rightarrow B$
 FACT Pierce-m2 : $(A \rightarrow (A \rightarrow B)) \rightarrow B$

cannot be verified by the constructive proof machinery of LINCOS (see also Chap. 3). That means that Peirce-m1 and Peirce-m2 are uncertain. Changing uncertainties into hypotheses is risky. We might want to see what can happen if this warning is neglected. So consider now by way of trial enriching the environment with

HYPOTHESIS Peirce-m1 : $((A \rightarrow B) \rightarrow A) \rightarrow B$.

Here A and B are supposed to be constants (in fact variables) of type Prop. It is of course not assumed that B is inhabited (the environment excludes it)—else the hypotheses would be trivial facts. The hypothesis Peirce-m1 is valid for any two propositions A and B .

The remarkable situation occurs now that, using Peirce-m1, B can be justified, *provided that A is inhabited*, i.e. $a : A$ exists. Formally it can be shown that

FACT one : B

is verifiable (Chap. 3). This statement can also be ascertained by the following reasoning: Observe $[H : A \rightarrow B] a : (A \rightarrow B) \rightarrow A$ and so $(\text{Peirce-m1 } [H : A \rightarrow B] a) : B$, i.e. $\text{one} = (\text{Peirce-m1 } [H : A \rightarrow B] a)$.

This result means that not only Peirce-m1 is verified, but also that *any* proposition B is verified! This is of course absurd: It is therefore forbidden to change the uncertainty of Peirce-m1 into a hypothesis, even though the environment is kept free of contradictions.

In somewhat the same way one can show that it is forbidden to change Peirce-m2 into a hypothesis. We are in fact left with only one admissible hypothesis:

HYPOTHESIS Peirce : $((A \rightarrow B) \rightarrow A) \rightarrow A$.

Uncertainty Instead of Falsity

One can imagine that an alien intelligent culture uses U (uncertainty) instead of F (falsity), satisfying the usual rules for conjunction and disjunction, but different ones for implication:

$T \rightarrow T = T$; $U \rightarrow T = U$; $T \rightarrow U = U$; $U \rightarrow U = T$.

In other words: $A \rightarrow B$ yields T if A and B are both T , or both U ; else U is yielded. An informal justification for this is: it is uncertain that an uncertainty can imply certainty; likewise it is uncertain that a certainty can imply uncertainty. The

resulting calculus is then an alternative propositional logic over truth and uncertainty. In this calculus we find the following:

$((A \rightarrow B) \rightarrow A) \rightarrow A$ is the sequence T U U T Peirce-u
 $((A \rightarrow B) \rightarrow A) \rightarrow B$ is the sequence T T T T Peirce-u1
 $(A \rightarrow (A \rightarrow B)) \rightarrow B$ is the sequence T T T T Peirce-u2

This table suggests using Pierce-u1 or Pierce-u2 as a hypothesis. But that would mean again that if $a : A$, then any proposition B can be verified. So even for an alien society using *uncertainty* instead of *falsity* and different rules for the implication connective, the hypothesis mentioned earlier is the only one to be kept:

HYPOTHESIS Peirce-u : $((A \rightarrow B) \rightarrow A) \rightarrow A$.

This result is due to the fact that constructive methods for verifying LINCOS theorems (called FACTS as we have seen) are consistent but not strongly dependent on propositional logic. At the same time there remains the uncertainty of knowing which kind of propositional logic is used (if at all) by an alien intelligent society. The good news is that it does not matter—what matters is that constructive means are used in their messages.

An Open Question for ETI

Invitation to ETI on the Internet is a project founded in the early years of the present century by Professor Alan Tough from Canada and consists of a communication addressed to any ExtraTerrestrial Intelligent society. It has more than 100 signatories from all over the world.

Supposing that we on Planet Earth are contacted by a seemingly alien intelligent society, claiming to be extraterrestrial, we would then wish to ascertain that the society is genuinely not terrestrial. One way of clearing up some of the uncertainty would be to send it a message containing a question—and upon receiving an answer judging the correctness of it. The question proposed is as follows:

“For propositions A and B , the implications

$(A \rightarrow (A \rightarrow B)) \rightarrow B$ and $((A \rightarrow B) \rightarrow A) \rightarrow B$

cannot constructively be verified as facts, i.e. theorems. However, it is persistently rumoured in the Galaxy that ([Gertjan Kamsteeg](#)):

For any given proposition B a certain intelligent civilization can think of, any such civilization can find a proposition A such that both expressions are valid.

We on planet Earth are interested in an explanation of the issue.
 Respectfully, ETI could you please explain this matter to us!”

Notes

Is there a certain correct answer to this question?

Have we got an uncertainty here?

What happens if an ETI has read the present book?

Reference

Thanks are due to my former Ph.D. student Gertjan Kamsteeg, for suggesting the original form of this question many years ago.

Appendix A Declaration of Principles Concerning the Conduct of the Search for Extraterrestrial Intelligence

Preamble

The parties to this declaration are individuals and institutions participating in the scientific Search for Extraterrestrial Intelligence (SETI).

The purpose of this document is to declare our commitment to conduct this search in a scientifically valid and transparent manner and to establish uniform procedures for the announcement of a confirmed SETI detection.

This commitment is made in recognition of the profound scientific, social, ethical, legal, philosophical and other implications of an SETI detection. As this enterprise enjoys wide public interest, but engenders uncertainty about how information collected during the search will be handled, the signatories have voluntarily constructed this declaration. It, together with a current list of signatory parties, will be placed on file with the International Academy of Astronautics (IAA).

Principles

1. *Searching*: SETI experiments will be conducted transparently, and its practitioners will be free to present reports on activities and results in public and professional fora. They will also be responsive to news organizations and other public communications media about their work.
2. *Handling candidate evidence*: In the event of a suspected detection of extraterrestrial intelligence, the discoverer will make all efforts to verify the detection, using the resources available to the discoverer and with the collaboration of other investigators, whether or not signatories to this declaration. Such efforts will include, but not be limited to, observations at more than one facility and/or by more than one organization. There is no obligation to disclose verification efforts while they are underway, and there should be no premature disclosures pending

verification. Inquiries from the media and news organizations should be responded to promptly and honestly.

Information about candidate signals or other detections should be treated in the same way that any scientist would treat provisional laboratory results. The Rio Scale, or its equivalent, should be used as a guide to the import and significance of candidate discoveries for the benefit of non-specialist audiences.

3. *Confirmed detections*: If the verification process confirms—by the consensus of the other investigators involved and to a degree of certainty judged by the discoverers to be credible—that a signal or other evidence is due to extraterrestrial intelligence, the discoverer shall report this conclusion in a full and complete open manner to the public, the scientific community, and the Secretary General of the United Nations. The confirmation report will include the basic data, the process and results of the verification efforts, any conclusions and interpretations, and any detected information content of the signal itself. A formal report will also be made to the International Astronomical Union (IAU).
4. All data necessary for the confirmation of the detection should be made available to the international scientific community through publications, meetings, conferences, and other appropriate means.
5. The discovery should be monitored. Any data bearing on the evidence of extraterrestrial intelligence should be recorded and stored permanently to the greatest extent feasible and practicable, in a form that will make it available to observers and to the scientific community for further analysis and interpretation.
6. If the evidence of detection is in the form of electromagnetic signals, observers should seek international agreement to protect the appropriate frequencies by exercising the extraordinary procedures established within the World Administrative Radio Council of the International Telecommunication Union.
7. *Post-Detection*: A Post-Detection Task Group under the auspices of the IAA SETI Permanent Study Group has been established to assist in matters that may arise in the event of a confirmed signal, and to support the scientific and public analysis by offering guidance, interpretation, and discussion of the wider implications of the detection.
8. *Response to signals*: In the case of the confirmed detection of a signal, signatories to this declaration will not respond without first seeking guidance and consent of a broadly representative international body, such as the United Nations.

Unanimously adopted by the SETI Permanent Study Group of the International Academy of Astronautics, at its annual meeting in Prague, Czech Republic, on 30 September 2010.

These revised and streamlined protocols are intended to replace the previous document adopted by the International Academy of Astronautics in 1989.

Appendix B Preliminaries

Evolution and Language

Darwinian evolution has led to large variety of living organisms and a multitude of systems for communication between them on Planet Earth. In his book *Darwins Ofullbordade* (Darwin's Unfinished), Bonnier (1997), Stockholm, the well-known broadly oriented Swedish neurophysician and novelist P.C. Jersild writes: "When Darwin claimed in 1859 in his theory of evolution that all now living creatures descend from just a few simple forms of life (that appeared four to three billion years ago), he had only superficial indications for that thesis. It is now known that the variety of species is due to differences in their *genetic maps*. Small differences in genetic codes may mean large differences between species, as between chimpanzees and humans (about 0.5 % difference in genetic code). After the branching resulting in the species that would become chimpanzees and those that developed into *homo sapiens*, it took five million years before the difference reached that value. In this way one can understand that the genetic clock ticks slowly" (translated shortened citation, AO). Interesting aspects of branching have been discussed extensively by Richard Dawkins in several books on palaeontology.

In order to obtain a perspective on the rates of Darwinian progress as it has occurred and is occurring on Earth, governed by the laws of nature, it is useful to compare the genetic clock with some astrophysical time scales. A period of five million years is in astronomical terms rather short: in that interval of time the Sun covers only 2 % of one revolution around the centre of our galaxy. This observation indicates already differences in rates of changes between earthly processes and those in an astronomical context. On Earth, biological (biochemical) processes can produce via genetic evolution complex *organisms* only slowly, in several hundred million years. In astronomy there are many evolutionary processes on time scales ten or more than ten times as large, for example in the formation and development of stars and galaxies. A normal medium-size star such as the Sun gets going as a proto star and evolves in about five billion years to its position in the main sequence of stars it has today. Stars are grouped together in *stellar systems* (galaxies),

themselves also evolving in time. Stellar systems form *clusters* arranged in a network of filaments in the universe. The universe itself, estimated to be 13.7 billion years old (measured from the primordial explosive event at the beginning of time), has evolved in this period to what we observe today: our own galactic system (the Milky Way) and its neighbours, the local cluster of galaxies and distant galaxies in various states of development, interstellar matter and intergalactic dark matter, and possibly also intergalactic dark energy.

The most important causal differences between evolutionary processes on Earth and those in astronomy are the prevailing circumstances. Life as we know and understand can occur only on the surface of a planet with moderate climate, surface and atmospheric conditions, admitting a rather narrow range of temperatures and a delicate balance of chemical substances—in short a rather restricted biosphere. Once life has established itself on a planet (in a very short time scale from the astronomical point of view), genetic evolution and mutations together with environmental effects evidently can produce complex *species*—and, following Jersild in the book mentioned before, also the social organization of groups within the species. Concurrently, languages and the use of them appeared on the scene. Jersild distinguishes five kinds of language: natural spoken and written language by people, gesture language (body language), the language of mental thought processes (also called *mentalese*) and the language of the genetic code based on the letters G, A, T and C (the building blocks of DNA) extended with rules of expression. Seamlessly one can fit into these kinds the language of music (see Chap. 15 of this book). These languages, and languages in general, are mutually strongly different in character and functionality. In the case of human languages their usage and development necessarily has been (and is) coupled to levels of intelligence and knowledge attained by mankind. As a result of increasing knowledge (in fact in philosophical terms “knowledge of the world”) there has been an enormous speed up in the development of societies, their organization and functionality—and in science. Scientific understanding of the world needs linguistic means for expressing insights and knowledge in order to be available to everybody.

Abstracting from these examples we note that for any *lingua* there must be a basic set of some kind of tokens together with rules for the formation of possible expressions (the *syntax* of language)—but also additional rules for the use and goal of these expressions (the *semantics* of language, supplying *interpretation*). Strictly speaking there is a commonly accepted restriction here: a language must admit written representations of expressions—not necessarily in digital form. Sufficiently developed human languages satisfy this requirement, cf. Tore Janson’s book *Speak: A short history of languages* (Oxford University Press, 2002).

As mentioned, language usage has an objective. Language serves as a means for communication between individuals or parties in societies, it can aid in the task of an individual forming opinions on matters at hand (via *mentalese*), but more generally it is needed for the spreading of knowledge. In genetics the (biochemical) rules of expression are responsible for the shape and functionality of individuals of a species. Language serves also to keep the vulnerability of living organisms under control using information coded in DNA, supplemented with fine-tuned, sophisticated expression rules. Even though humans are usually not fully aware of the way the

genetic code with associated processes operates, this particular linguistic background governs the line of life of each individual.

Astrolinguistics Conceptually

The rather broad conceptual framework of language and its usage sketched in the previous section leads to the philosophical question whether in astronomy, i.e. in the cosmos of stars and stellar systems, interstellar matter and voids, dark matter and dark energy, there are phenomena that conceptually seem to be influenced by what can be termed *astrolinguistics*. In view of the prevailing physical circumstances in the cosmos as we understand them, the question might be split into two parts.

First, a point of inquiry could be: can one designate a system of processes in the universe, governed by specific rules, evidently leading to some goal. One case is that of astrophysical processes leading naturally to the formation of stars (even from a more fundamental cosmological point of view) with planets orbiting many of them. As a result of planeto-physical processes in and on these planets together with developments in their seas, atmospheres and geology, life might come into being on some of them. A necessary condition for that to happen is, according to prevailing understanding, that such a planet must orbit its sun in a well-defined habitable zone—not too distant and not too near its sun. The rules governing the processes leading to life are the physical and chemical laws of nature. Biological properties supervene on these laws, *cf.* Chap. 11. Until now only one instance of such a planet is known (the Earth) but there is ample evidence by now to assume that the physical circumstances that led to the case of the Earth are *not* unusual. Note in passing that our universe satisfies the *comprehensibility principle*, stating that of all possible universes, only those in which observations can be made by intelligent beings, are understandable for them. Case proved because we humans are beginning to understand our Universe, the building stones—elementary particles and their interactions—and evolutionary processes. Packets of energy in various forms can be considered to be the basic tokens of a linguistic principle on a cosmic scale. The syntactic rules are those governing interactions between the tokens. In particle physics the comprehensibility principle prevails as well. In physics, astrophysics and cosmology the observational instrumental techniques for investigating the tokens have evolved tremendously since mankind started looking at nature and the skies systematically.

In the second place one might inquire whether linguistics of natural languages on Earth is some sort of derivative of a general cosmic principle, valid for all living intelligent beings in the Universe. It would seem that a straightforward analysis of this matter is out of reach for the moment. One could argue that our languages have been developed and have evolved for our own use, within the limitations of the human brain, not on the basis of a cosmic principle. However, an important feature of natural languages is the fact that they are able to explain their own rules governing “well-formedness” of expressions. Children learn their native language effortlessly, without being consciously aware of grammar and formation rules for expressions.

Early in life, they grasp the semantics of their vocal expressions by trial and error. Later on in life they learn to handle non-trivial complexities of their language explained in that language itself. Human languages are able to interpret (explain) themselves—they admit in fact *self-interpretation*. Is this linguistic feature part of or derived from a general cosmic principle?

Suppose that we earthlings some time from now in the future discover that we are not alone in the universe and that there exists in the galaxy a planet with a society of intelligent beings. If we would receive a message unmistakably emitted by those *aliens*, how are we to interpret the message? If it would contain some means for self-interpretation we might be able to get a relatively quick start in understanding some of the message content. More important, if this feature is provably present in such a communication, a strong ground would have been achieved for the assumption that the concept of self-interpretation is a general cosmic linguistic feature. The feature itself is an essential part of what can be understood to be *astrolinguistics*.

The above discussion leads to an important point: should we on Earth decide to start a project aimed at *Communication with ExtraTerrestrial Intelligence* (CETI) and agree on transmitting a message into the Galaxy, then we should use a *Lingua Cosmica* (abbreviated to LINCOS) based on the simplest possible grammatical structures. At the same time the expressive power of the *lingua* should be sufficiently large to express information we deem useful to transmit. These requirements should be met because our messages must be conceived in such a way that they fulfil two goals: it must be possible for *aliens* to recognize that the message is of a *linguistic* kind and the message is meant to be understandable for them—in fact after the non-trivial task of decoding the stream of digitized information has been completed. In addition extra information should be included in the LINCOS text informing receivers that the system admits self-interpretation (in accordance with the general astro-linguistic principle outlined above). These requirements, however, are part of a more general issue. For CETI we need to identify and exploit the use of an astrolinguistic *common ground*—a conceptual system which all intelligent symbolic species in the Universe can be assumed to share with one another. The search for a common ground as meant here is like the search for the Holy Grail. The existence of it is uncertain and it might even prove to be an ideal never completely reachable. We can, however, strive to get a better understanding of the issues involved—and designing a linguistic system for interstellar communication based on logic (as presented in this book) can contribute in this respect. Should we find the right base for CETI, a kind of looking glass, it would give us a clear view of the road to be taken: like Anceaux's glasses gave a Papuan in New Guinea a clear view of his and our world (see Fig. 1).

Anceaux's glasses not only symbolize the need of obtaining a clear view of the possibilities of CETI. Likewise they help us to realize the existence of some aspects of the fundamental problems facing the development of a *Lingua Cosmica* for CETI research:

- Enormous cultural and linguistic differences between human societies and those of intelligent beings elsewhere in the universe are to be expected.
- Communication over interstellar distances in real time is impossible as far as we know the laws of nature, excluding tachyonic velocities; therefore the possibilities of “testing” the effectiveness of LINCOS are extremely limited.

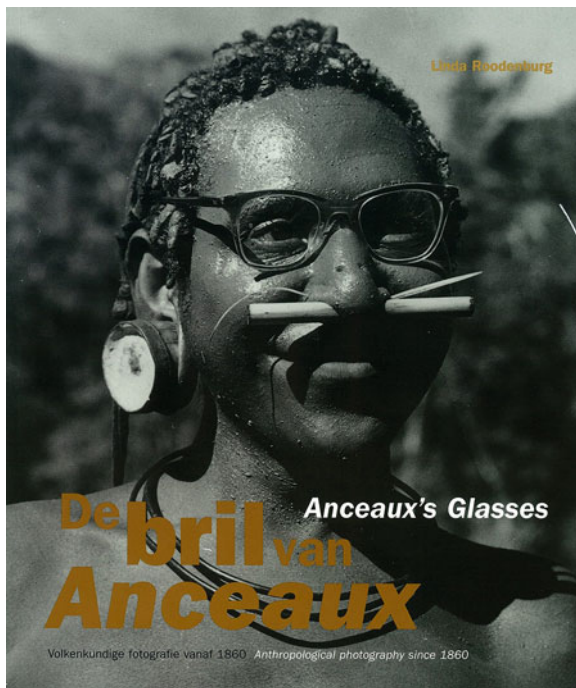


Fig. 1 *Anceaux's glasses*. Reproduced by permission. The National Museum of Anthropology, situated in Leiden, The Netherlands, organized in the year 2002 the exhibition “Anthropological Photography from 1860” and one of the pictures displayed was the one shown in this figure. Linda Roodenburg’s text for the exhibition catalogue mentions: “Professor Anceaux, the well-known Dutch linguist and anthropologist from Leiden University took part in 1959 in the last Dutch expedition to New-Guinea. He gave his spectacles to a Papua—the picture shown was taken on that occasion. Anceaux’ glasses symbolize the Western view on ‘the others’ and the Papua wearing the glasses is a symbol for ‘the others’, those who in this way can look back at us”. (Translation by the author of the present book). Dr. Anceaux was a former colleague of the author

- The possibility of effective CETI might lie beyond our horizon at this point of time; after more than 50 years of SETI with the largest radio telescopes of the world, we have not seen any signs of (intelligent) life in the neighbourhood of the Sun and beyond.

Logic as Common Ground

The choice of logic as the base of a *Lingua Cosmica* is motivated by the view that logic can be considered to be a reasonable and useful common ground for interstellar communication between galactic symbolic species. One cannot expect a species without the power of logical reasoning to be able to interpret an interstellar meaningful

message. Two questions then present themselves immediately: what kind of logic to choose and in which manner to use it.

The first question is related to the purpose of logic *ipso facto*. The common denominator of all applications is correct reasoning over abstractions of reality, so that argumentation on the basis of tools of logic leads to reliable results. At the same time logic is useful for clarification. For these purposes many kinds or modalities of logic have been developed. The propositional calculus is a basic one. The syntax is simple: assertions are combined into expressions using the “and”, “or”, “implication” and “negation” connectives. The meaning of the expressions, i.e. the semantics of the calculus, is given by the well-known truth tables. Assertions, however, are restricted to abstractions of individual entities, and in this calculus one cannot express abstractions over a class of objects satisfying some predicate. For achieving that predicate calculus has been developed, using the “all” and “exists” quantifying operators (quantors) in addition to the propositional connectives. It is therefore a stronger logic. Because of these operators the syntax is more extensive. The semantics of this calculus is rather complicated.

In order to appreciate the latter somewhat, suppose all entities in some collection have a property in common expressed by a predicate. Then the assertion that there exists an entity with the mentioned property is not correct unless such an entity existed already or is constructible with instruments of the logic. Observations of this kind have lead to the development of *constructive* logic. Expressions in this modality of logic are provided with types, either by introductory declarations or by simple construction and reduction rules. The existence problem mentioned above is resolved by the declaration of a constant entity of the right type, i.e. that of the collection.

Using ingredients from constructive logic, abstracts of message content are elegantly expressed. The modality chosen admits simple grammatical syntax, sequential notation, decidability of conclusions (since they must be constructible in a finite number of steps) and large expressive power. Interpretation of expressions in this logic by recipients presents a separate problem. There are in fact two (intertwined) issues involved here: the need for a signature identifying the modality and the explanation of the semantics of the system employed. Both of these are non-trivial matters. For solving these questions either self-interpretation can be used or else recourse must be taken to instruments exterior to the logic. In interstellar message composition, the use of natural language for the latter purpose is not applicable. Natural language is capable of self-interpretation, but only understandable for observers who know the language sufficiently well. In our case we describe at a meta level, i.e. outside of the language, the logic contents of a message, written in sentences and collected into larger units. As a side effect, features of the language employed might become understandable, but providing linguistic insights is beyond the goal of the research reported here (Fig. 2).

The image symbolizes some fundamental problems confronting research in the field of Communication with ExtraTerrestrial Intelligence. Of course there will be immense differences (also culturally) in operational linguistics as used between terrestrial human societies and as used between extraterrestrial alien ones—but the

Fig. 2 *Blue Planet Calling*. Reproduced by permission. Created by artist C. Bangs, New York, in 2003, especially for Astrolinguistics



magnitude of them cannot be estimated. Can the somewhat mystical face of the woman, embedded in terms of logic, but also projected on our Blue planet, help in understanding each other? In any case the image will hopefully trigger interest in our messages, whether or not formulated in the new LINCOS.

Research in Astrolinguistics

The previous section illustrates that multidisciplinary research on aspects of life in the Universe, its origin, existence and evolution on Earth and elsewhere, connects in various ways with linguistics in a general sense. Species as they develop and evolve will devise linguistic means for communication. Since communication between intelligent species on spatially separated planets may need to overcome large distances, one is even *a fortiori* concerned with astrolinguistics. Intra planetary communication in *real time* might be feasible but only in the case of multiple planets supporting intelligent life and orbiting around the same star. On planet Earth research *ipso facto* in astrolinguistics has been carried out in several fields:

- Search for ExtraTerrestrial Intelligence (SETI), detection of linguistic information in low-noise electromagnetic radiation from the Universe.
- Fundamental principles of communication between mutually different and totally unknown intelligent species—over interstellar distances.
- Universal semantic machines.

- Coding and decoding algorithms.
- Construction of interstellar messages.
- Creation of a *Lingua Cosmica*.

Moreover the following topics can be considered to be related to the field of astrolinguistics as well:

- Astropsychology—studies of Darwinistic or other kinds of development of intelligent, symbolic (i.e. linguistic) types of life on planets more or less similar to Earth.
- Astroarcheology—searches for artefacts produced by intelligent life in the galaxy and intentionally or accidentally left behind, “forgotten;” studies of linguistic principles as templates for the development of life elsewhere.

The scientifically challenging topics in the broad field of astrolinguistics are evolving strongly since the first SETI searches and have already led to a number of interesting projects and results. In the SETI Institute in Mountain View, California, there is the productive research group *Interstellar Message Construction* (project leader: Prof. D. Vakoch). The interesting research by Dr. John Elliott in England in the field of semantic machines has produced valuable and useful analyses—while



Fig. 3 *Alexander Ollongren. Conceptualizing LINCOS. Photographed by Dap Hartman (2000). Reproduced by permission. The author had already seriously started developing the new *Lingua Cosmica* around the time this photograph was taken in Leiden in the Netherlands. The term Astrolinguistics for the surrounding scientific discipline was coined in an article by the author in Wikipedia (2010)*

further development is to be expected. The present author's second-generation *Lingua Cosmica* detailed in this book represents a new step in designing linguistic systems for interstellar communication. This project will certainly be followed by further research—some possible lines are suggested in the present book. Astroarcheology and astropaleontology are appearing as completely new disciplines, witnessed for instance by the work of Dr. Kathryn Denning in Canada and the Fermilab Dyson Sphere Searches directed by Dr. Richard Carrigan in the USA. Work in these fields is sure to attract and stimulate new research in the humanities. Finally the important large and broad fields of bioastronomy and astrobiology need to be mentioned. These fruitful areas of research forcefully emerged in the last two decades, have already attracted many researchers from various disciplines, and is fully blossoming right now (Fig. 3).

Appendix C History

*I miman fick vi in att det finns liv
på flera hall.
Men var ger miman ej besked om.
Det kommer spår och bilder, landskap och fragment av språk,
som talas någonstans, men var.[0]
Harry Martinson [1], ANIARA [2], En revy om människan i tid
och rum [3], 1956
Poem 6, verse 1*

Background

The orientation of the present monograph is multidisciplinary. It is concerned with universal aspects of linguistics (here referred to as astrolinguistics), applied logic and especially conceptual non-technical issues in the field of possible message exchange (communication) between intelligent species (or information processing artefacts) in the Galaxy using a *Lingua Cosmica*. The (astro) linguistic system advocated for that purpose is based on formal logic. The modality of the logic used is constructive, supplying the design of the LINCOS for interstellar communication with a solid foundation. As an introduction to the enterprise and problems of designing a *Lingua Cosmica* we supply the following historical remarks.

More than 50 years ago Elsevier North-Holland Publishers in Amsterdam brought out LINCOS, *Design of a Language for Cosmic Intercourse*, Part I [Freudenthal 1960], written by the well-known prominent Dutch mathematician Dr. Hans Freudenthal († 1991) of Utrecht University in The Netherlands. Upon getting acquainted with the book many years ago, the present author, educated as a mathematical astronomer in the Netherlands and Sweden, interested in computer science and logic, was almost immediately fascinated by the conceptual problems in designing a language for communication between mutually alien intelligent species in the universe. The book was the first one (and is the only monograph until now) on this topic. Freudenthal's brilliant design of the *Lingua Cosmica* has concepts from

mathematics and some logic as well at the core. It was published in Elsevier's series *Studies in Logic and the Foundations of Mathematics*. From the way it was written it is evident that the renowned mathematician enjoyed this excursion into the area for him unfamiliar. That impression was enhanced during the one and only meeting the present author had with Prof. Freudenthal. It occurred during the Dutch mathematician's congress at the Mathematical Institute of Leiden University in The Netherlands in the late eighties of the last century. Colleagues of the present author, having attended in the 1960s Dr. Freudenthal's public lectures on the unusual subject, have confirmed this impression. The work on LINCOS was, however, incomplete. Part II, planned to be devoted to the description of societal aspects of humanity, was perhaps partly conceived but was never published. It is unclear whether topics in this field have been studied more than only cursorily before, during or after the publishing of Part I.

An interesting, comprehensive and in-depth review of Freudenthal's book written by Bruno Bassi was to be found a few years ago in Wikipedia. The paper is entitled *Were it perfect, Would it Work Better? Survey of a Language for Cosmic Intercourse* [5]. As that article presents unusual and illuminating perspectives not only on the book itself but also on philosophical aspects of Freudenthal's undertaking, some relevant quotations are incorporated into the present historical resumé. The author is thankful for the permission received from Dr. Bassi for quoting him extensively here. In the opening paragraph of the review Bassi stresses the point that there is:

... a sort of paradox that is implicit in the whole enterprise, an oscillation between a formalist and a communicative trend that makes Lincos a hybrid experiment from the point of view of the design of a perfect language.

Bassi also remarks:

... Lincos is a very peculiar educational (gedanken-) experiment. Usually, linguistic education takes place through the use of a language already known to the learner (as in case of learning by adults) and/or in real-world contexts in which people smile, frown, gesticulate, point at objects, and in which the learner can observe other speakers' behaviour and get feedback from them (as happens to children and anthropologists). On the contrary, in the ET case we can rely neither on a known language nor on an extra-linguistic context. All we can do is to speak pure Lincos. The language is to be taught through the language itself, used one-way in an absolutely pure fashion.

There are some further important points in Bassi's article, quoted here:

... 'Decoding Lincos would be an easy job' (Freudenthal 1974:1828). Provided that the ETs who are receiving it fulfill certain requirements. A basic requirement concerns their technology: they must be able to receive radio signals and to measure their duration and frequency. In order to understand Lincos texts they should be humanlike with regards to mental states and communicative experiences. In particular, if they are to understand the initial part of the program, they should have intuitive arithmetical conceptions somehow similar to ours. This may seem a strong assumption. However, given that we have to start off with some universally understood topic, the choice of natural numbers arithmetic seems to be, after all, quite a reasonable one. Then, our ETs must of course have some sort of language of their own. It may be completely different from our languages, but its handling of context, of presupposition and of implication should be essentially the same as what we are accustomed to. ... Anyway, it is *not* requested that ETs already know all the things we are telling

them about. A lot of mathematics as well as ‘behavioral rules’ are learnable through Lincos itself once an agreement about the fundamentals has been reached.

... Dr. Freudenthal explicitly introduces his language as a step towards the design of a ‘*characteristica universalis*’. Due to the progress achieved in this century by formal logic, we should be much closer to such a result than Leibniz, for instance, was. The only trouble is the difficulty of choosing a starting point. We need to start with a “concrete, sharply defined and narrow problem” (p. 12). The problem of communicating with ETs should serve as such a starting point.

... A man decides to build a perfect language. The philosophical tradition in which he places himself is one that pursued formalization, both as a requirement to be fulfilled by the artificial languages it produced and as a deep principle to be posited underneath the surface structure of natural languages. He claims that the formal instruments at his disposal are adequate to the task. Then, in the actual design of the language, he applies his formal devices only to a few specific syntactic features, and for the rest he relies on the structure of natural languages. People usually build perfect languages in order to override traditional ‘unsatisfactory’ features of natural languages, such as the fact that they are subject to nonsense, ambiguity, lexical and grammatical irregularity, context-dependency. Yet, many of these features are still somehow present in Lincos. Why did Dr. Freudenthal, who has not at all a naive approach to this sort of issues, let things go this way?.

... The idea of applying achievements from symbolic logic to the design of a complete language is deeply linked to a strong criticism towards the dominant twentieth century trend of considering formal languages as a subject matter in themselves and of using them almost exclusively for inquiries about the foundations of mathematics. ‘In spite of Peano’s original idea, logistical language has never been used as a means of communication ... The bounds with reality were cut. It was held that language should be treated and handled as if its expressions were meaningless. Thanks to a reinterpretation, ‘meaning’ became an intrinsic linguistic relation, not an extrinsic one that could link language to reality’ (p. 12).

... In order to rescue the original intent of formal languages, Lincos is bound to be a language whose purpose is to work as a medium of *communication* between people, rather than serve as a formal instrument for computing. It should allow anything to be said, nonsense included. In Lincos, ‘we cannot decide in a mechanical way or on purely syntactic grounds whether certain expressions are meaningful or not. But this is no disadvantage. Lincos has been designed for the purpose of being used by people who know what they say, and who endeavour to utter meaningful speech’ (p. 71).

... As a consequence, Lincos as a language is intentionally far from being fully formalized, and it has to be that way in order to work as a communication tool. It looks as though the two issues of communication and formalization radically tend to exclude each other. What Lincos seems to tell us is that formalization in the structure of a language can hardly generate straightforward understanding.

... Our Dr. Freudenthal saw very well this point. ‘there are different levels of formalization and ... in every single case you have to adopt the one that is most adaptable to the particular communication problem; if there is no communication problem, if nothing has to be communicated in the language, you can choose full formalization’ (Freudenthal 1974:1039).

... But then, how can the solution of a specific communication problem ever bring us closer to the universal resolution of them all? Even in case the Lincos language should effectively work with ETs, how could it be considered as a step towards the design of a *characteristica universalis*? Maybe Dr. Freudenthal felt that his project needed some philosophical justification. But why bother Leibniz?.

... Lincos is there. In spite of its somewhat ephemeral ‘cosmic intercourse’ purpose it remains a fascinating linguistic and educational construction, deserving existence as another Toy of Man’s Designing.

Freudenthal, Hans 1960 *Lincos—Design of a Language for Cosmic Intercourse*, Amsterdam, North Holland 1974 “Cosmic Language”, in T. A. Sebeok (ed), *Current Trends in Linguistics*, vol 12, The Hague: Mouton, pp. 1019-1042

SETI

The publication of Freudenthal's book, now only available antiquarian, coincides with the beginning of an epoch: the start of international projects in the Search for ExtraTerrestrial Intelligence (SETI). Rather remarkable is that Martinson's book ANIARA was published a few years earlier. That visionary, interesting book pre-udes in the form of poetry on the immense difficulties that can be expected in the field. In the more than 50 years elapsed since then several aspects around SETI have been cleared up [Ronald D. Ekers et al 2002]. Modern astrophysical research has revealed that it is not to be expected that *real-time* interstellar communication will be possible. There exist indeed numerous solar type stars of about the same age as the Sun in the habitable zone of our Galaxy, many planets orbiting stars have been found but so far none of them are earth-like (astronomers expect to find this type of planets in the near future). Exiting discoveries are to be expected as soon as the observational techniques will permit analyses of planetary atmospheres. However, there is the fact that our nearest neighbours (whether or not harbouring intelligent species) are on the average at a distance of scores of light years away. Laws of physics forbid the transmission of information with tachyonic velocities (exceeding the velocity of light). So no highly developed technological society can expect to "get in touch" directly with another one in the Galaxy. Nevertheless such societies (and ours too) can be assumed to be not only interested in transmitting information about themselves to whoever is "listening", but are in fact even putting some effort in doing so. One reason might be because they could be inquisitive (in the sense of striving to acquire knowledge—as our species is by nature) and wish to know about "the others". For both the purposes of transmission and reception of messages, a rather sophisticated linguistic system for interstellar communication is needed. In the author's view Freudenthal's design, brilliant as it is, is outdated now. The design of a new system should satisfy at least some basic properties:

- Linear notation and simple syntax.
- Clarity of expression (self-interpretation of messages).
- Rich contents of messages, redundancy.
- Possibility of structuring and sizing messages.

In addition the system should be able to describe not only static relations but also dynamic processes. This extremely important capability is discussed in some detail in the present book, in fact in the separate PART V—devoted to the representation of various kinds of processes in the proposed new LINCOS. In addition this part presents an opening to the matter of representing and using aspects of (computer) information processing programs in the system.

It is not unreasonable to assume that an intelligent species receiving an interstellar message unmistakably bearing a linguistic signature will put automatic information processing machines at work to do the decoding and some of the interpretation. We would do the same thing! Therefore messages should be large-sized and contain a large amount of redundancy.

Multi-Level Approach

The writer of the present new book, belonging to a generation younger than Prof. Freudenthal's, has since the late fifties been exposed intensively to basic concepts in computer science: low-level assembly programming systems, procedural (ALGOL-like) and functional (LISP-like) and logic programming languages, besides formal languages and automata—and the mathematical/logical *lambda* calculus. All of our computer programs in low-level languages *had* to be written in the early days of computer science in linear notation, preferably in prefix form, i.e. the Polish version. Years later, those experiences supplied the author with the idea that Freudenthal's abundant use of super- and subscripts and the numerous *ad hoc* agreements (resulting in a rather unwieldy notation), might be simplified with some effort by using ingredients from “modern” theories on computer programming. At the same time it was realized that the overall purpose of Freudenthal's work might be achieved in a better way if one *abstained* from using just a single level in interstellar message construction—that of the *Lingua Cosmica* itself.

Thus an idea was born: messages meant for interstellar communication with extra-terrestrial intelligent societies should be essentially *multi-level*, they should consist in part of a (large) text in some natural language supplemented with annotations in a formal system at another level. In that case the other level has the role of a *meta* level in which descriptions about the contents of the basic level are available. The basic level can but need not necessarily contain only text. Pictures supplemented with text but also music could be placed there. Using *formal logic* at the core of a *Lingua Cosmica* enables one to describe not only the logic contents of texts in such messages but also the definitional framework (comparable with an environment created by the vehicle of declarations in computer programming). Seen from this starting point the proposal in this treatise is in fact a linguistic *system* to be used for interstellar communication.

In the wake of this view Freudenthal's idea of using mathematical notions as a central core in the language was abandoned. Instead it was realized that sophisticated *type* declarations and *type* notions from proposition and predicate logic should be given central positions. Logic reasoning about textual contents can be (and are in the present setting) shifted to the mentioned *meta* level. If mathematical reasoning of some sort is needed to explain properties, that can be done in yet another level.

When the author got well acquainted with the French Coq [Gérard Huet et al 1999] implementation of the calculus of constructions with induction (CCI) based on the typed lambda calculus, it was realized that the basics of that proof system provide a suitable vehicle needed for the design of a new linguistic system for interstellar communication. The specific formal logic on which the new system is based, has by its nature rich declarative and expressive powers. It seemed not to be a good idea to translate Freudenthal's expressions into terms in constructive logic. A better idea was to use CCI and the underlying type theory in a totally different way and circumvent at the same time some of the problems present in the first *Lingua Cosmica*. One of these is that that *lingua* is very suitable for expressing mathematical relations, but

much less useful for describing structural aspects of human societies. As mentioned before, Freudenthal's unpublished Part II was meant to deal with aspects as these. In the late 1980's, Prof. Freudenthal, during the mentioned conversation with the present author, gave the impression of having lost interest in the project.

A predominant requirement for the present author's undertaking was, right from the beginning, that a message meant for an extra-terrestrial civilization should in some way carry information for the interpretation of the formal language employed—or methods for achieving this goal. So aspects of auto-interpretation have been incorporated and worked out in the system. The new *Lingua Cosmica*, also referred to as LINCOS, developed and described in the present monograph rests on concepts of CCI formulated in terms of type theory. Moreover it uses linear notation. Several computer implementations of this particular logic are available. Some of these have been used for verifications of lemmas in this book. Therefore occurring facts (i.e. lemmas in the form of term expressions in LINCOS) in the book are guaranteed to be correct—a remarkable, important aspect. The elimination machinery employed is rather different from that in the Coq system. The notation, largely based on the conventions of type theory, has been adapted in order to improve readability. The set of primitives is kept very small. As a result of this the dynamics of verifications become transparent. The requirement of obtaining an “easy to handle” (linear) notation is met as well. All of these aspects are explained in detail in the book.

The study of interstellar communication requires *ipso facto* an interdisciplinary approach. It is concerned with cosmology, astrophysics, -chemistry, -biology, but also with information processing, linguistics and coding theory besides concepts in mathematics and logic. So the underlying central themes of the book are in fact situated in the broad context of astrolinguistics. Within that context the new *lingua* is used for describing in a concise and interpretable manner a selection of physical reality as we humans experience it. For that purpose abstractions of reality, and more in particular logic static relations occurring in reality, are modelled. They are represented mostly as strictly logical forms. It appears, however, that modelling of more involved dynamic relations needs extra means of reasoning—as in cases treated in Part V. That can be done by arbitration utilizing a separate level.

An Example

Some of the easier to understand ideas behind the proposed LINCOS are illustrated below by a simple example (see for details Chap. 7). We use for that in the present notes one of the easier well-known syllogisms of the Greek philosopher Aristoteles (384–322 BC). The important concept of Aristotelian syllogisms has survived more than 2,000 years of development in logic and has been enormously influential. At one level (say the basic level) one might use the following example of an Aristotelian basic syllogism:

all Humans are Mortal and *all* Greeks are Human
so *all* Greeks are Mortal.

This kind of expression, called *Barbara* by mediaeval scholars because of the three occurring symbols *all* (logic quantifiers), is evidently representative of a whole class of logic expressions.

In Aristotelian logic there are four mutually distinct basic classes of expressions (cf. Chap. 7). A suitable *environment* is needed in order to describe in LINCOS at another level (the *meta* level, but it is in a way also a *deep* level) the logic contents of the particular expression shown above. It consists of a universe of discourse *D*, and moreover humans, mortality and Greeks introduced as *types* using LINCOS notation:

CONSTANT *D* : Set. (universe of discourse)

CONSTANTS Human, Mortal, Greek : $D \rightarrow \text{Prop}$.

(maps representing subjects and predicates)

These are declarative items, specifying the environment needed.

Set is a predefined type and *D* of type Set is introduced.

Prop is also a predefined type, distinct from Set.

If *a* and *b* are abstract representations of logic propositions (assertions) then

a, *b* : Prop, i.e. they have type Prop. The type of Human is $D \rightarrow \text{Prop}$, i.e. a (mathematical one to one) mapping or function. Note that Human, Mortal and Greek have the same type. So if *x* has type *D*, then the functional application of Human to *x*, written (Human *x*) has type Prop but also (Mortal *x*) and (Greek *x*) have the same type Prop.

The syllogism itself is written in the spirit of Aristotelian logic as a lemma, stating the non-elementary type of Barbara

FACT Barbara :

(ALL *x* : *D*)

((Human *x*) $\rightarrow$ (Mortal *x*)) $\wedge$ ((Greek *x*) $\rightarrow$ (Human *x*)) $\rightarrow$

(Greek *x*) $\rightarrow$ (Mortal *x*).

Here $\rightarrow$ denotes logic implication and $\wedge$ is used for logic conjunction. Above fact is easily understood by humans (and perhaps ETI)—it hardly needs verification. But in order to explain it in a logical sense it needs a proof. The present book explains in general how facts are verified in a formal sense *within the system itself*. In above case Barbara can be shown to be equal (in Leibniz's sense) to a constructed lambda form (see Chap. 7).

Suppose that in one of our messages for interstellar communication the Aristotelian syllogism as shown (e.g. in the form of a text file) is embedded. In it are then above declarations and the fact Barbara of the non-trivial type shown—perhaps augmented with a proof (verification) of the fact. A recipient of this message wishing to decode and understand the contents of this fragment faces several non-trivial problems. In order to simplify the issue we suppose that it is recognized that one of the natural languages spoken on Earth (unknown to the receiver) is used at one level, and that the logic structure of the sentence is explained at another level with the help of terms in a logic system. The recipient knowing (propositional and predicate)

logic, upon analysing the incoming signals (text), recognizes that a constructive form is used, and will discover soon the meaning of ALL (universal quantification), and the connectives $\rightarrow$ (implication) and $\wedge$ (conjunction). He/she/it might immediately add to these $\vee$ (disjunction) and $\sim$ (negation), as well as Ex (existence), absent in this example. Furthermore the items Human, Mortal and Greek appear in the text as well as in the deep structure (and in the proof of the fact). All of this is helpful for the decoding and interpretation problem.

However, we may well assume that small-size messages will be unintelligible for recipients or their information processing artefacts (i.e. ETI), despite much effort from their side. In order to effectively provide help for the interpretation problem, messages should therefore contain much or very much redundancy. Thus, as a part of a more extensive message in LINCOS, one could include and formulate at least several examples of all four basic Aristotelian syllogisms. They need not necessarily be in the form available in Chap. 7 of this book, because those are meant to be informative for human readers. In the present monograph the author has developed the necessary formalizations in terms of the linguistic system proposed. Thus each of the four basic syllogisms is formulated as a fact, verifiable within the framework of the language itself. This (general) powerful aspect of the system may prove to be one of the important keys for decoding purposes.

There are other instruments as well for explaining in our messages the structure and conceptual set-up of the LINCOS system. These are assumed to be feasible and effective because the design of the system is based on extremely simple grammatical rules. One could *e.g.* use music at the basic or even third level with annotations to the score in the LINCOS language, see Chap. 15. Also useful for this purpose is pictorial information (possibly at again another level) as for instance available on the famous anodized gold plaques on board the Pioneer 10 (launched in 1972) and Pioneer 11 (launched in 1973) unmanned space crafts. The contents of pictures can (partially) be described in LINCOS, see also Chap. 14. Using multiple levels coded information of these kinds could be included (simultaneously) as well.

Note: some material in the present book is based on the author's contributions to the international congresses of the International Astronautical Academy from 1998 onwards. The author has retained the copyrights.

References, translations

[0]

In miman it is found that there is life
in several places.

But where the computer does not reveal.

Traces and pictures arrive, landscapes and fragments of language,
spoken somewhere, but where. (transl. by the author)

[1] Harry Martinson (Sweden) 1904–1978, received the Nobel Prize for literature in 1974

[2] Anlara, a space ship, has left planet Earth for a voyage into unknown space. Martinson described (in 1956) in 103 poems the fate of the people aboard, the crew and *miman*, the advanced intel-

- ligent computer. From the introduction: Aniara is concerned about everything we personally do not master, but which is part of our lives and belongs to us. (transl. by the author)
- [3] A story about man in time and space. (transl. by the author)
- [4] H. Freudenthal 1960 *LINCOS, design of a Language for Cosmic Intercourse*, Part I, North-Holland Publishing Company
- [5] The article appeared in the volume “Le lingue perfette”, edited by Roberto Pellerey, Versus—Quaderni di Studi Semiotici 61/63, 1992, and is available at <http://www.brunobassi.it/scritti/lincos.html>
- [6] SETI 2020 *A Roadmap for the Search for Extraterrestrial Intelligence*, Editors: Ronald D. Ekers, D. Kent Cullers, John Billingham, Louis K. Scheffer, SETI Press, MountainView, Calif. 2002
- [7] Gérard Huet et al. 1999 *The Coq Proof Assistant*, early version, INRIA, France

Appendix D A Gentle Introduction to Lambda and Types

Intention

Two of the lesser known theories in mathematics and logic are the λ *Calculus* and *type theory*. In the astrolinguistic setting of this treatise, more in particular for the design of a system for interstellar message construction discussed, both of these are of prime importance. The new *Lingua Cosmica* described in this book, aimed at interstellar communication, is in fact a *linguistic system*, and these two theories supply the mathematical foundation of the proposed lingua. Therefore it is appropriate that necessary background information on the use of the lambda- and type concepts is supplied. That is the purpose of the present chapter. It is set apart as an appendix because the theories associated with these concepts are rather unusual even though they are in essence quite easily understood. Because of the central position they occupy in the present work, they are discussed here (albeit in a gentle way) in some detail. Readers not supposed to possess any prerequisite knowledge in the relevant fields, are enabled in this manner to get easily acquainted with the main aspects of the theories.

Pillars

The *Lingua Cosmica* (LINCOS) and its use in astrolinguistics as described in the present treatise rest on two main pillars: the *Lambda* (or λ) *Calculus* and a *Calculus of Constructions* including inductive definitions (and therefore sometimes referred to as CCI). In this treatise the Calculus of Constructions used is often referred to simply as CC, even though the germane concept of induction is for some applications (*e.g.* recursion) extremely important. These calculi (see also details described in PART I) with roots in intuitionist logic in turn utilize a number of aspects of the (typed) λ Calculus, unfortunately not well-known outside of mathematics, logic and theoretical computer science. In the formalism of the λ Calculus, functions and functional applications are represented in an unusual way. In the typed version of the λ Calculus

each entity is supplied with an abstract *type*. Types, typing and type checking occupy central positions in CC as well. Therefore it is appropriate to discuss in the present appendix some of the concepts and notations figuring in the λ Calculus. There is no need to give a comprehensive discussion of that particular calculus in this book because only a restricted set of expressions is actually used in LINCOS. Concepts, on the other hand, are discussed in the following sections in somewhat more detail. The standard volumes on the Lambda Calculus and its various ramifications are the important books by Prof. H. Barendregt [Barendregt 1984, 1992].

Lambda

The λ Calculus was originally developed by Alonzo Church around 1940 as a logico-mathematical system for formalizing fundamental aspects of mathematics. A remarkable follow-up around 1960 was the definition by John McCarthy of the *List Programming Language* based on this calculus. LISP became eventually the prototype of functional computer programming languages. In contrast to the ALGOL-like (or procedure oriented) languages, these languages are based on the notion of functions (or maps). That idea was inherited from the original λ Calculus without types. McCarthy's LISP functional programming system, which became famous, is also untyped. The author's experience in LISP programming has influenced the development of LINCOS as described in this book, but only indirectly because the Calculus of Constructions uses *typed* λ Calculus, see also PART I of the present treatise. In the present chapter we review both the untyped and typed calculi, albeit briefly, because they provide together a ground and solid pillars for CC and therefore a fortiori for our LINCOS.

We must remark here that the notation employed in CC and LINCOS resembles strongly the notation used in the present chapter, but it is not exactly the same. However, notation in LINCOS is kept as close as possible to the one used in the following sections. Another aspect is that newer computer implementations of the Calculus of Constructions tend to drift away from the conventions of the λ Calculus in view of demands emanating from the area of applications. In our design we have refrained from that tendency since we need a general purpose approach, and also because of requirements brought up by one of our important design objectives: the possibility of self-interpretation of LINCOS. Still deviation from the notation of this section could not be avoided, because of the requirement that LINCOS terms must be expressed in a linear notation. See also the remarks in the POSTSCRIPTUM of this book.

Untyped λ Calculus

In the present section we use arithmetical operators and integer constants, even though they do not figure prominently in LINCOS. This choice is motivated by the fact that concepts in this calculus are easily explained in that way. At the same time

we emphasize that this does not mean that arithmetic's *ad fortiori* should be embedded in messages for Extra Terrestrial Intelligence (ETI).

Beginning with the calculus without types: let $f, g, \dots, x, y, \dots$ be *variables* and let there be a collection of *constants*, e.g. the integers, the Booleans or something else, with denotations written in some notational system. Standard *operators* of arity 1 or more (1 or more arguments required), such as the arithmetical $\sqrt{}$ ("square root", arity 1), $+$ ("addition", arity 2), $-$ ("subtraction", arity 2), $\times$ ("multiplication", arity 2) and $/$ ("division", arity 2), or logical $\wedge$ ("conjunction", arity 2), $\vee$ ("disjunction", arity 2), $\rightarrow$ ("implication", arity 2) and $\sim$ ("negation", arity 1) can be considered to be constant (or fixed) *functions*. A variable can also be considered to be a function, but then of arity 0. The context in which a variable appears shows which set of constants is the relevant one. The essence of the untyped calculus is that the relevant sets are not made explicit. *Expressions* in the calculus are built using variables, constants and functions, formally

$$\begin{aligned}\text{expr} &::= \text{constant} \mid \text{variable} \mid \text{function} \\ \text{function} &::= \text{expr expr} \mid \lambda \text{ var}^+. \text{expr}\end{aligned}$$

This is a context-free generative grammar (in the Chomskyan sense) describing the linguistic deep structure of the λ Calculus. Note the extreme simplicity of the grammatical rules. The notation used here is derived from the report defining the syntax of the Algorithmic Language ALGOL-60, by the computer scientists J.W. Backus et al., and P. Naur (Editor) in 1960. The token $\mid$ denotes a choice, expr expr denotes functional *application*, and $\lambda \text{ var}^+. \text{expr}$ is a so-called *lambda-abstraction*. var^+ indicates a sequence of one or more than one variable.

The (extremely simple, but complete) grammar given above is abstract (in a linguistic sense), showing form structure only. In order to obtain surface structures the constants, variables and functions must be represented by *denotations* in some way. It is seen that a function is either an application of one expression to another or a lambda-abstraction. As above grammar supplies syntactic form only, comprehensive semantics are needed to give *meaning* as well to expressions and functions. Doing this comprehensively for the λ Calculus is a formidable task and lies far beyond the scope of the present treatise. Instead we shall discuss the necessary fundamental semantic concepts figuring in the calculus using illustrations. First expressions, abstraction and binding rules are explained.

Expressions representing mapping from a domain to a codomain are written in prefix notation. The simplest λ -abstraction has the following form

$$\lambda x. \text{expr.}$$

In this case only one variable (x) occurs in the λ -abstraction and the domain of the mapping is the domain of the variable. In the λ -form shown, the dot is followed by expr , called the *body* of the abstraction; it is an expression for a map from the domain of x to the codomain. So expr will in general be dependent on the variable x , *bound* by the λ and with as the *scope* of it the body of the abstraction. Note that x can but need not occur in the body—this is evidently an aspect of prime importance.

The body can contain further λ -abstractions with local variables. If such variables are introduced in the body, the usual *scope rules* in the theory of high-level programming languages apply—we will not go into this in any detail. The λ -abstraction above is a way of defining a mapping without giving it a name—this is also important. The map itself is written by convention in the operator prefix notation, i.e. operators are written first followed by its arguments. So we might have

$$\lambda x. (+x\ 31)$$

representing the concept “adding thirty one”. The variable x ranges over some domain, admitting an interpretation of the $+$ operator.

Next *conversion rules* are explained. Above example is not the only representation of the concept is concept “adding thirty one”. Consider for instance

$$\lambda y. (+y\ 31)$$

representing the same operation. So we have

$$\lambda x. (+x\ 31) = \lambda y. (+y\ 31)$$

the equality sign expressing that the two expressions are the same, not literally but representing the same mapping. The one and only difference in their appearances is that the variable x has been renamed to y (or the other way around). This process of renaming is called α -conversion. An example of a simple expression with two bound variables, showing another α -conversion is

$$\lambda x, y. (+x\ y) = \lambda y, x. (+x\ y).$$

A λ -expression representing a function can be *applied* to an argument to yield a result. Consider the following example of the concept of functional application

$$\begin{aligned} (\lambda x. (+x\ 31))\ 21 &= +\ 21\ 31 \\ (\lambda x, y. (+x\ y))\ 21 &= \lambda y. (+21\ y) \end{aligned}$$

Note the replacement rule used here: 21 must be substituted for x (the first formal argument) not for the second, y . The case of two applications is illustrated by

$$(\lambda x, y. (+x\ y))\ 21\ 31 = (\lambda y. (+21\ y))\ 31 = +\ 21\ 31$$

It is seen that the underlying association (in fact functional composition) is from left to right. Further one can say that applying a λ -expression to arguments entails a form of evaluation. The mechanism illustrated by these examples is called β -conversion. Note that β -conversion is not able to change $\lambda x, y. (+x\ y)$ into $\lambda x. (+x\ 31)$. The following example uses only β -conversion

$$(\lambda x, y, z. (+x\ y))\ 21\ 31 = \lambda y, z. (+21\ y)\ 31 = \lambda z. (+21\ 31) = +\ 21\ 31$$

The last step is justified because z does not occur in the expression $(+ 21 31)$.

One additional conversion rule, the so-called η -conversion needs also to be mentioned.

Let f be a function of arity at least 1, independent of a given variable x . Let $(f x)$ be the application of f to x . Write $f x$ for this case, a simple one as far as brackets are concerned. Consider the equality $(\lambda x. f x) y = f y$ due to β -conversion (and because f does not contain x). This means that the equality $\lambda x. f x = f$ is justified. This remarkable result is called the η -conversion rule. Note the equality $(\lambda x. f x) x = f x$.

Here are some examples showing useful conventions.

Because brackets in function application associate to the left (one can say that they *cluster* on the left-hand side), they need not always be written:

$$\begin{aligned} + 21 31 &= ((+ 21) 31) \\ + (+ 21 31) 41 &= (+ ((+ 21) 31) 41) \\ ((f g)x) &= (f g)x = f g x, \text{ functional composition,} \\ &\quad \text{first } f \text{ is applied to } g \text{ then the result to } x \\ f(g x) &= f(g x), \text{ first } g \text{ is applied to } x \text{ then } f \text{ to the result.} \end{aligned}$$

Sometimes it is necessary to give a name to a λ -expression representing a function (especially in view of applications in LINCOS). So for the first of above examples we might write

$$\text{DEFINE } f := \lambda x. (+ x 31).$$

This feature is mandatory in the case of functions defined in terms of themselves: i.e. when we need recursive (or inductive) definitions. The standard example of this is the definition of the product $n! = 1 \times 2 \times 3 \times \dots \times n$ for any natural number $n \geq 0$, using another declarator

$$\text{INDUCTIVE fac} := \lambda n. (= n 0) \rightarrow (\text{fac } 0) \mid \sim (= n 0) \rightarrow (\times n (\text{fac } (-n 1))).$$

The above is a recursive (or inductive) formalization of $(\text{fac } n)$ in terms of $(\text{fac } 0)$ and $(\times n (\text{fac } (-n 1)))$, where $(\text{fac } 0) = 1$ can be added as an “afterthought”. The body of the λ -form contains two clauses. The expressions $(= n 0)$ and $\sim (= n 0)$ have the role of induction hypotheses. The vertical stroke $|$, a *separator*, is used to keep the hypothesis apart from one another. Under a recursive evaluation of fac with some natural number as argument, both hypotheses are evaluated successively. These aspects return in the discussions on CC.

Note: the tokens **DEFINE** and **INDUCTIVE** are examples of the so-called *declarators*.

Consider now the following alternative definitions, the first inductive, the second not.

INDUCTIVE $\text{Fac} := \lambda \text{fac}', n. (= n\ 0) \rightarrow (\text{fac}'\ 0) | \sim (= n\ 0) \rightarrow (\times n (\text{fac}' (-n\ 1))) \text{Fac}.$
 DEFINE $\text{H} := \lambda f, n. (= n\ 0) \rightarrow (f\ 0) | \sim (= n\ 0) \rightarrow (\times n (f (-n\ 1))).$

This implies the equality $\text{fac} = \text{H fac}$. In other words the map H applied to fac yields fac . This means that fac is a *fixpoint* of the map. One expects that the fixpoint can be computed, for instance that $3!$ can be evaluated to $(\times 6 (\text{fac}\ 0))$. That is done as follows using β -conversion several times, not on Fac but on fac .

$$(\text{fac}\ 3) = (\times 3 (\text{fac}\ 2)) = (\times 3 (\times 2 (\text{fac}\ 1))) = (\times 3 (\times 2 (\times 1 (\text{fac}\ 0)))) = (\times 6 (\text{fac}\ 0)).$$

The last step requires evaluation of the multiplication operator over the natural numbers. The computational facility using β -conversion shown here is in principle available in LINCOS as well. Its usefulness is in general, however, limited. An exception is the case of representing processes (Chap. 17). Alternatively we might extend LINCOS with concepts of symbolic computation (computer algebra), see PART VI.

Typed λ Calculus

We proceed now with a discussion of the *typed* λ Calculus. We mentioned in the previous paragraph that an expression as $\lambda x.(+x\ 31)$, considered to be a function, can be given an argument in the range of the variable, some number but certainly not a Boolean. If the range of x is the set of natural numbers, the requirement can be expressed by adding the *type* of x in the expression, by changing it into

$$[\lambda x : \text{nat}].(+x\ 31).$$

In this way x is supplied with the type nat , the type of natural numbers (distinct from the set of natural numbers). The semicolon is used to formalize the relationship “has type” or “is of type” (see also Chap. 1). Types generally are abstract and can be structured. The constituents of above expression should then also have types. To begin with, take an important step by stating the type of addition

$$+ : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}.$$

This is because the map designated by $+$ is a binary operator expecting two arguments of type nat and delivering a result of type nat . This means for example

$$\begin{aligned} (+21) &: \text{nat} \rightarrow \text{nat} \\ (+21\ 31) &: \text{nat}. \end{aligned}$$

Since it has been explained that the λ expression can be applied to an argument of type nat to yield a result of type nat , it should be no surprise that the type of the complete expression is

$$[\lambda x : \text{nat}]. (+x \ 31) : \text{nat} \rightarrow \text{nat}.$$

In the typed λ Calculus everything should be typed. Consider for example the definition

$$\text{DEFINE } f := [\lambda x : \text{nat}]. (+x \ 31).$$

The type of f is evidently also $\text{nat} \rightarrow \text{nat}$. As a result $(f \ 21) : \text{nat}$, and using β -conversion it is seen $(f \ 21) = (+ \ 21 \ 31)$. In agreement with this, note that $+$: $\text{nat} \rightarrow \text{nat} \rightarrow \text{nat}$, so $(+ \ 21 \ 31) : \text{nat}$. In view of later requirements in the Calculus of Constructions, consider another example, but now using the quantifier ALL. Let g be declared by

$$\text{HYPOTHESIS } g : (\text{ALL } y : \text{nat})(+y \ 31).$$

The token HYPOTHESIS is also a declarator. The type of g designated in this way is different from the type of f . In addition we have the normal form $(\text{ALL } y : \text{nat})(+y \ 31) : \text{nat}$, with arity 0 for g . See H. Barendregt [Barendregt 1984] for a comprehensive discussion on normal forms. Substituting 21 for y we find $(g \ 21) : (+ \ 21 \ 31)$, where $(+ \ 21 \ 31) : \text{nat}$. As $(+ \ 21 \ 31) = (f \ 21)$, we find rather remarkably, $(g \ 21) : (f \ 21)$.

These examples show that a map can be defined as a λ term, but also as a type, written as a hypothesis using the universal quantifier. We have here a *principle of choice*. If a definition is chosen, β -conversion can be used for evaluation purposes, if a hypothesis defining a type is declared, evaluation is evidently not possible. The principle characterizes the difference between the introduction of an entity being by specifying its type and by defining it to be equal to another entity. If two entities have the same type, they are said to *type check*, but they need not be equal. In the next examples the declarator VARIABLE is used.

$$\text{VARIABLE } a : \text{Prop}.$$

$$\text{DEFINITION } I : \text{Prop} \rightarrow \text{Prop} := [\lambda x : \text{Prop}]. x.$$

$$\text{Note}(I \ a) : \text{Prop and } (I \ a) = a.$$

Sometimes we use the declarator DEFINITION instead of DEFINE.

$$\text{VARIABLE } a : \text{Prop}.$$

$$\text{HYPOTHESIS } g : (\text{ALL } y : \text{Prop}) y \rightarrow y.$$

$$\text{Note}(\text{ALL } y : \text{Prop}) y \rightarrow y : \text{Prop and } (g \ a) : a \rightarrow a.$$

Next consider the case of recursively (inductively) defined types (functions). As an example consider the recursive map `fac` discussed above. In introducing types in the inductive definition we must see to it that `fac` is assigned the type $\text{nat} \rightarrow \text{nat}$, of arity 1, and in addition we arrange that the inductive clauses are typed. Let the latter be given names `h1` and `h2`. The definition of `fac` written as a parametrized function becomes then

INDUCTIVE fac $[\lambda n : \text{nat}] : \text{nat} :=$
 $\text{h1} : (= n 0) \rightarrow (\text{fac } 0) \mid \text{h2} : \sim (= n 0) \rightarrow (\times n (\text{fac } (-n 1)))$.

The form $[\lambda n : \text{nat}]$ is moved to the left in order to introduce the *selectors* h1 and h2. Note that the defined fac requires an argument of type nat and delivers a value of type nat, so the requirement $\text{fac} : \text{nat} \rightarrow \text{nat}$ is fulfilled. The constants h1 and h2 should *not* be assigned types globally, i.e. outside the definition of fac. Here are the types, using n , inherited from the definition of fac

$\text{h1} : (\text{ALL } n : \text{nat}) ((= n 0) \rightarrow (\text{fac } 0))$.
 where $(\text{ALL } n : \text{nat}) ((= n 0) \rightarrow (\text{fac } 0))$: nat, the normal form.
 $\text{h2} : (\text{ALL } n : \text{nat}) (\sim (= n 0) \rightarrow (\times n (\text{fac } (-n 1))))$.
 where $(\text{ALL } n : \text{nat}) (\sim (= n 0) \rightarrow (\times n (\text{fac } (-n 1))))$: nat, the normal form.

In passing we note that

$[\lambda n : \text{nat}] ((= n 0) \rightarrow (\text{fac } 0)) : \text{nat} \rightarrow \text{nat}$
 $[\lambda n : \text{nat}] (\sim (= n 0) \rightarrow (\times n (\text{fac } (-n 1)))) : \text{nat} \rightarrow \text{nat}$.

The possibility of referring to induction hypotheses outside definitions is useful and will be exploited in applications of LINCOS. So, as expected the inductive clauses are correctly typed.

Combinators

We leave the set of natural numbers aside in this section and suppose instead that we have available a basic collection of entities called Set. The collection is not interpreted in the set theoretical sense and no special set theoretical operators such as membership, intersection or union over the collection are assumed. However, abstract mappings over Set and conglomerates will be used, but not interpreted (i.e. we do not associate mathematical objects with those). The combinators to be introduced here are useful for definitions of functions in terms of basic ones. But in this section we define them in order to illustrate with examples the phenomenon of *type checking*, of prime importance in applications of LINCOS. Note that a function of arity $n \geq 1$ applied to one argument yields a new function of arity $n \geq 0$ and of another type.

Consider the following definitions:

DEFINE I := $[\lambda x : \text{Set}] . x$
 DEFINE K := $[\lambda c, x : \text{Set}] . c$
 DEFINE S := $[\lambda f : \text{Set} \rightarrow \text{Set} \rightarrow \text{Set}; g : \text{Set} \rightarrow \text{Set}; x : \text{Set}] . (f x (g x))$

So that

$$\begin{aligned} I &: \text{Set} \rightarrow \text{Set} \quad K : \text{Set} \rightarrow \text{Set} \rightarrow \text{Set} \\ S &: (\text{Set} \rightarrow \text{Set} \rightarrow \text{Set}) \rightarrow (\text{Set} \rightarrow \text{Set}) \rightarrow \text{Set} \rightarrow \text{Set}. \end{aligned}$$

In the following examples of type checking we make use of the rule that brackets in function application associate to the *left* so that they are not written.

I is called the *identity function* because

for any $x: \text{Set}$, $I x: \text{Set}$ and $I x = x$, using β conversion.

For any $c, d: \text{Set}$, application of the defined K function yields this

$K c d: \text{Set}$ and $K c d = c$, using β conversion.

$I c$ and $K c d$ *type check* and they are also *equal*. This might be called the *chameleon effect*, as it is much like the ability of the reptiles referred to of changing their colour while retaining their identity.

Another kind of examples is obtained by applying S to K , justified because the first argument of S must be of type $\text{Set} \rightarrow \text{Set} \rightarrow \text{Set}$. Before application S has arity 3, so

$$SK : (\text{Set} \rightarrow \text{Set}) \rightarrow \text{Set} \rightarrow \text{Set of arity 2}.$$

Since $I: \text{Set} \rightarrow \text{Set}$, SK can be applied to I , so with β conversion

$$SKI : \text{Set} \rightarrow \text{Set}.$$

Next, using the definition of S , consider equalities

$$\begin{aligned} SK &= [\lambda g : \text{Set} \rightarrow \text{Set}; x : \text{Set}]. (K x (g x)) \\ SKI &= [\lambda x : \text{Set}]. (K x (I x)) = [\lambda x : \text{Set}]. (K x x) = [\lambda x : \text{Set}]. x = I. \end{aligned}$$

So SKI *type checks* with I and at the same time $SKI = I$. It is seen that SKI and I also are subject to the chameleon effect.

For any maps $f: \text{Set} \rightarrow \text{Set} \rightarrow \text{Set}$, $g: \text{Set} \rightarrow \text{Set}$ and for $x: \text{Set}$ we have

$$\begin{aligned} S f g x &: \text{Set}, \text{ and on the other hand} \\ g x &: \text{Set}. \\ f x &: \text{Set} \rightarrow \text{Set}. \end{aligned}$$

$f x(g x): \text{Set}$, so $(S f g x)$ and $f x(g x)$ are of the same type (they *type check*) whereas equality is not concluded.

Various chapters in the main body of the present book supply an impression of the prominent role of type checking in the design of LINCOS.

References

- [1] H.P. Barendregt *The Lambda Calculus: its syntax and semantics*, North Holland Publ. Comp. (1984)
- [2] H.P. Barendregt *Lambda calculi with types*. Handbook of Logic in Computer Science, Vol. 2, Oxford Sci. Publ., pp. 117–309, Oxford Univ. Press, New York (1992)

Appendix E Postscriptum

Views in Retrospect

Intention

In these “afterthoughts”, written after completing the main parts of the manuscript, we endeavour to embed the undertaking which resulted in the present treaty, in a wider context. In addition some thoughts are formulated on applications of the linguistic system not discussed hitherto. Also briefly considered are possible ways and means for further development of LINCOS, as well as the validation issue. Finally the significance of developing a *Lingua Cosmica* is analyzed.

Universals in Language

Chomsky’s paradigm on properties of natural languages, for a long time leading in general linguistics, includes the (strong) assumption that there exist universals in languages. They would be abstractions of structural properties common to all languages used on planet Earth. This idea would make sense under the supposition that humans are born with the ability in the developing brain to recognize universals of this kind and make use of them in learning how to handle languages operationally.

It seems clear indeed that children already at very young age are able to recognize auditive and visual *patterns* supplied by the environment they live in. How patterns such as these are processed and reprocessed (because there is a dynamic aspect as well) is a subject for research—but that some kind of abstract representation of environmental information is stored in the (developing) brain seems an unavoidable conclusion. As a consequence of this humans might be supposed to have the capability of recognizing linguistic patterns in kinds and diversity in natural language, and storing representations of them in their brains. Such patterns could be specific for a given language, not necessarily subjected to a Chomskyan generative

grammar, and need not be truly universal. They can, however, be considered to be intimately connected to, in fact represent the *signature* of a given language. In that case the concept of signatures of languages together with the human capability of recognizing them as such would be the cornerstones of language processing by humans, be it in an auditive sense or in reading textual representations.

This, however, need not necessarily be the only way in which (linguistic) information processing could be carried out by extraterrestrial intelligent beings or their information processing artefacts (supposing that they exist). Apart from that aspect, any linguistic system for interstellar communication should possess a specific signature over possible expressions, not only for the purpose of distinguishing them from random noise, but especially because in that way an aid for the interpretation of language content is supplied. We have seen in Chap. 16 that the LINCOS system as presented in this book possesses specific signatures.

Universal Structures

Are there universal structural properties of some kind in systems for information exchange used (spoken and/or written) by intelligent beings and societies elsewhere in the Universe, within or between them? The existence of some kind of means (languages or otherwise) for information exchange between intelligent beings need not be questioned. If elsewhere in the Universe languages are involved, we do not know whether they are more or less structurally similar to—or perhaps entirely different from natural languages used by humans on planet Earth. It seems to be impossible to falsify the existential question as far as languages are concerned—simply because the universe is too large for an exhaustive search. In addition there seems to be no evident useful inductive principle available in order to arrive from evidence in a restricted part of the universe to a valid conclusion in the large. But also verification will be hard to obtain, without quite a bit of luck!

In the chapter PRELIMINARIES we mentioned the important role a *common ground* can be expected to play in possible CETI. In this book we are using logic for that purpose. Let it be mentioned here that we do *not* assume that the complete set of tools at the base of LINCOS should be recognized or understood more or less immediately by receiving ETI. From messages containing an (possibly large) amount of redundancy, one can imagine that receivers could eventually reconstruct (or even guess) the basic ingredients. A process such as this might be somewhat similar to the way humans reconstruct memories. Neuro-medical studies seem to indicate that reconstruction is based on the processing of many large or even small fractions of impressions of happenings stored in the brain. The processing time scale involved in the case of ETI attempting to decode our messages might be considerable from the human point of view.

In view of the foregoing, in the discipline of astrolinguistics one is not explicitly interested in the development and use of a *lingua universalis* for interstellar communication. One pursues a much more modest goal: research on the capabilities and possibilities of a rather restricted linguistic system for (effective) information exchange, a *lingua cosmica*—in our case one with the possibility of validation (as

explained in Chap. 10). A legitimate scientific question is then: what characteristics should such a system have, what kind of signature should it possess, what universal structures are presupposed? It seems clear from the discussions in the PRELIMINARIES what it preferably should *not* possess: a linguistic signature similar to those we know and study in natural languages on Earth. Because of this, the term *lingua (cosmica)* might be considered to be in some way a bit of a misnomer. Yet, we feel that it is justified to use this term in view of common usage of expressions such as “the language of mathematics”, “the language of logic”, but also “formal languages and automata”, and “programming and algorithmic languages”.

Since all expressions in any astrolinguistic communication system must necessarily be digitized and represented by linear strings of symbols organized in streams, the question of characterization can be reformulated: what digitized patterns are or should be distinctive in a *lingua cosmica*?

Partially an answer of a rather general nature has been given already for the particular communication system LINCOS explained in the present treatise. The signatures of the system, as discussed in Chap. 16, result from the basics of the system, the formation rules for terms, the use of a context-free abstract grammar inclusive recursion for the deep structure, and deeply rooted expressions. Important formation rules are those governing the creation of accessible environments by means of declarations. Then there are the rules governing the appearance of terms and restricting possible expressions. Examples are: the functionality of delimiters, restrictions on declarations (usually deterministic, except in PART VI of this book), the occurrence of free and (λ -)bound variables, scope rules, a strict agreement on the use of pre-, post- or infix notation—preferably not intermixed, and the use of inductive structures.

The importance of patterns in LINCOS is evident throughout the book, but is especially clear in the formulation of Aristotelian expressions already mentioned in the history part and fully worked out in Chap. 7. We have examined and explained in detail there the basic pattern called a *predication*, often used in language. Elementary examples of these (containing a singular) are:

- Socrates is a Greek, all Greeks are human, so Socrates is human.
- Socrates is a Greek, no Greek is an animal, so Socrates is not an animal.

Provided many examples of these kinds are contained in messages, including a fair amount of redundancy, predications as such should be recognizable for ETI as special *forms*. The various kinds of predications in the form of Aristotelian syllogisms are reviewed in a short section towards the end of this chapter.

Future Studies

The aspect of validation, mentioned in Chap. 10, deserves further discussion. One form of validation is already present in the *Lingua Cosmica* developed in the present treatise, because, as we have explained, a form of self-interpretation is built in: lemmas, theorems (called facts) are verifiable within LINCOS itself. A presupposition is that for a given application the operational environment (a stage) is sound. In

order to guarantee that, or at least to assure that the terms in a given environment are non-contradictive, the first thing to do is to consult Wittgenstein's Theatre. But that is conceivably impractical in most cases because of the sizes, i.e. the number of terms the relevant objects generally consist of.

What perhaps could be done as well is replacing the environment by an equivalent stage formulated in an existing computer implementation of constructive type logic. Here a problem might be the designing of an equivalent stage, given one in LINCOS, see [Ollongren 2012]. The equivalence should also be verified, how? Matters such as these can hopefully be cleared up by future studies. Assuming Freudenthal's LINCOS to be the first generation and ours the second, future studies certainly might lead to a third, new-generation *Lingua Cosmica*, one in which open questions such as these are tackled.

On the Interpretation Problem

LINCOS as a linguistic system rests on several pillars. One of them is the multilevel character of the system. The level carrying actual expressions in the formalism proposed is mostly used to annotate information contained in other levels where other formalisms and expressive systems are used (expressions in natural language, possibly pictorial information, music). So LINCOS terms are interpretations of other information—but the other way around can also be claimed: information in another level can be useful for clarification of LINCOS. This view brings up the question whether LINCOS texts *an sich* could be supplied with interpretations. Could interpretation be carried out by automata? LINCOS terms describe logical relations—not in the first place operations and actions as for instance computer programs do. Therefore one is *not* in the first place interested in an operational semantics using e.g. interpreting automata (see [Ollongren 1974]). Suppose that ETI's artefacts are to do the interpretation of LINCOS texts. They would have to do a kind of relational analysis in order to arrive at an interpretation. Hopefully the abstract signature of LINCOS expressions (or collections of expressions) will prove to be helpful in indicating that concepts from the lambda-calculus and constructive logic are involved. For the purpose of attaining more or less complete interpretations, generally knowledge of the state of affairs ("knowledge of the world") is needed. This is one of the serious difficulties the designers of messages from Earth for ETI are faced with: which knowledge of the world and how much of it should be transmitted?

Self-Interpretation, -Knowledge and -Reflection

Apart from the more general matter of interpretation and in a way more basic is the matter of auto-interpretation. Natural languages are able to explain their grammatical structures, expressions and rules forming them, by themselves—i.e. a natural

language can explain itself—fortunately for our youngsters learning their first and perhaps at the same time another language. One of the cornerstones of Chomskyan theory is the use of recursion in describing the grammatical structure of a natural language. The important feature of self-explanation should preferably be present in any cosmic language as well. The vehicle for achieving this in the case of LINCOS described in the present treatise is the proof machinery for facts—sometimes using the concept of logical induction, see Chap. 2, but always without recourse to external means, see Chap. 13. The basic principle is that facts are verifiable within the system itself. In essence this is due to the general fact that expressions in the system are so to say explained by their types, more in particular: are shown to satisfy correctness criteria, in terms of the system primitives themselves.

Next there is another matter: the question whether LINCOS possesses a form of self-consciousness. In terms of design criteria of the system we can split the question in two parts. To begin with the system evidently “knows” all information stored in the environment extended with proven facts and dynamically extensible. “Knowing” means here that the system has this information accessible and can use it. What the system does with available information depends on tasks given to it. A simple task could be: “Prove Fact F”, where the type of F is in the environment, but the proof not (yet). Once the system constructs a proof, its knowledge base is enlarged with that proof and the system is able to refer to it (self-consciousness). Note that LINCOS is not designed to create tasks on its own—in this respect there is no self-consciousness.

However, in a different respect there is a form of self reflection. This results from the fact that the system is designed as a multilevel system. At some level it may contain a text, and at a (in a way deeper) level the text is interpreted (often by way of the construction of proofs, verifications of statements). But we can of course imagine the use a level containing *programs*, themselves consisting of tasks, such as simple ones as in the example above. Tasks can also be of a self-reflecting type, for example “reconsider” a sequence of steps. In this way the system would be able to “know” or “consult” the information embedded in programs, and descriptions of “what to do”, as well. In the wake of this the system could be able to learn from experience, i.e. using knowledge acquired in decision making. This might be achieved by means of communication between levels. In the present treatise, aspects as these have not been worked out in detail—that could be a task for a new-generation LINCOS in the future.

Another aspect is the question whether LINCOS is able to communicate with a program not only in the sense of extracting information *from* it but by a facility for adding information *to* it in such a way that the program is modified. By verifying (proving) a theorem the environment is enriched—and we can enquire whether a program could be enriched by verified facts. Intimately connected to this question is what interpretation should be given to programs in the LINCOS environment [Ollongren 1974]. A new generation of LINCOS in the future might address these matters.

Finally the question may be asked: why not use one of the computer implementations of formal proof systems—instead of developing LINCOS *ab initio*?

Doing that would mean loss of transparency. One would then be concerned with three layers. One containing information in some language more or less formalized. Another containing a “translation” to clauses in the proof system and then a third layer with the verifications in yet another formalism. In LINCOS the last two aspects are integrated in one transparent system: a linguistic system based on logic (and only logic).

Supervenience Revisited

We have seen that the purpose of the LINCOS system is to describe aspects of reality in a broad sense. So a LINCOS text is a kind of coded imprint of something in reality—one could also consider LINCOS descriptions to be projections of reality. The philosophical question discussed in this section is whether LINCOS descriptions supervene reality (or reality subvenes LINCOS). The roots of the concept of supervenience lie in philosophical body–mind discussions—material and immaterial aspects of life. Supervenience is discussed *e.g.* in D.J. Chalmers [Chalmers 1996], *The Conscious Mind*. A citation from that book:

The notion of supervenience formalizes the intuitive idea that one set of facts can fully determine another set of facts. The physical facts about the world seem to determine the biological facts, for instance, in that once all the physical facts about the world are fixed there is no room for the biological facts to vary. (Fixing all the physical facts will simultaneously fix which objects are alive). This provides a rough characterization of the sense in which biological properties supervene on physical properties.

One could argue—if some *situation* in reality is fixed as a set of facts and fully described in LINCOS, then there is no room for an alternative description, it is determined by the real facts of the situation. So LINCOS would supervene on reality—provided that complete descriptions can be attained. Practice shows that descriptions are usually incomplete—only parts of a situation are covered. If those are the main parts of a situation in some sense, the point of view that reality subvenes on LINCOS is reasonable.

In order to get a grip on the matter of supervenience from a logical point of view and formalize the concept one can consider an alternative and discuss a *system of situations*. In that case the following often quoted rather curious definition of supervenience can be used:

- If the properties of a system (of situations, each characterized by a set of facts) can be subdivided in two classes: B-properties (*high level*, *e.g.* biological properties) and A-properties (*low level*, *e.g.* physical properties), then B-properties supervene on A-properties if no two possible situations are identical with respect to their A-properties while differing in their B-properties.

Rather remarkable here is that the vagueness in this description can be removed by formulating an equivalent definition in terms of LINCOS—avoiding double negation. For that purpose we prefer to use the simplified form:

- B-properties supervene on A-properties if for any two possible situations identical with respect to their A-properties, this implies that they are identical too with respect to their B-properties.

Let S : Set represent a system of situations and let $A:S \rightarrow \text{Prop}$ and $B:S \rightarrow \text{Prop}$ represent the A-properties and the B-properties of the system. Further let $s1, s2 : S$ be two distinct situations. $(A\ s1) : \text{Prop}$ is then the A-property of situation $s1$, $(B\ s1) : \text{Prop}$ is the B-property of $s1$. In the definition above the notion of properties being identical occurs. So we need a definition of equality suitable for the case under consideration. It is inspired by the equality function inductively defined in Chap. 2, i.e.

```
INDUCTIVE Eq [X : Prop; x : X] : X → Prop := Eq-intro
: (Eq X x x).
Eq-intro : (X:Prop; x : X) (Eq X x x).
```

In view of this we introduce the equality function we need for the systems mentioned above, such as this

```
INDUCTIVE Eqs [X : Prop; Y : Prop; x : X; y : Y] : X → Y
→ Prop :=
Eqs-intro : (Eqs X Y x y).
Eqs-intro : (X : Prop; Y : Prop; x : X; y : Y) (Eqs X Y x y).
```

Suppose that for situations $s1$ and $s2$, we have for the A-properties $a1 : (A\ s1)$, $a2 : (A\ s2)$ and $b1 : (B\ s1)$, $b2 : (B\ s2)$ for the B-properties. In that case

```
(Eqs (A s1) (A s2) a1 a2 ) : Prop
```

expresses that the A-properties of $s1$ and $s2$ are the same, and

```
(Eqs (B s1) (B s2) b1 b2 ) : Prop
```

expresses that the B-properties of $s1$ and $s2$ are the same. We have now arrived at the stage where supervenience, and as a bonus subvenience can be introduced as hypotheses.

```
HYPOTHESIS supervenience : (Eqs (A s1) (A s2) a1 a2 ) → (Eqs
(B s1) (B s2) b1 b2).
HYPOTHESIS subvenience : (Eqs (B s1) (B s2) b1 b2 ) → (Eqs
(A s1) (A s2) a1 a2).
```

Discussion

Returning Chalmers' book we quote some relevant passages below.

It should be stressed that the logical supervenience is not defined in terms of deducibility in any system of formal logic. Rather, logical supervenience is defined in terms of logically possible *worlds* (and individuals), where the notion of a logically possible world is independent of these formal considerations. This sort of possibility is often called 'broadly logical' possibility in the philosophical literature, as opposed to 'strictly' logical possibility that depends on formal systems.

At the global level, biological properties supervene logically on physical properties. Even God could not have created a world that was physically identical to ours but biologically distinct. There is simply no logical space for the biological facts to independently vary. When we fix all the physical facts about the world—including the facts about the distribution of every last particle across space and time—we will in effect also fix the macroscopic shape of all the objects in the world, the way they move and function, the way they physically interact. If there is a living kangaroo in this world, the *any* world that is physically identical to this world will contain a physically identical kangaroo, and that kangaroo will automatically be alive.

In the present chapter we have shown that supervenience can be defined in terms of constructability in a system of formal logic, using LINCOS as a carrier. This aspect is a sideline in discussions on truth in LINCOS, see Chap. 11. More important is the observation that the LINCOS system supervenes logically on reality.

Aristotelian Syllogisms Reviewed

Examples of *predications* are (to be a) Greek, (to be) human, (to be) animal. Subjects can act as predications. A single subject (*e.g.* Socrates) is a singular.

Let D be the universe of discourse and let at least one object be of type D , $d : D$, i.e. D is the case. Let an individual be represented here by the constant d . We can use d to represent for instance Socrates. In that case Socrates exists—or historically existed! Let S and P be subjects and predicates (in the form of predications) both over D , represented by maps $\text{is-}S$ and $\text{is-}P$ from D to Prop . In LINCOS we have

```
CONSTANT D : Set.
CONSTANT d : D.
CONSTANTS is-S, is-P : D → Prop.
```

A short review follows of the cases **Aps**, **Asp**, **Eps**, **Esp**, **Ips**, **Isp**, **Ops** and **Osp** as discussed in extenso in Chap. 7. This review is meant to display Aristotelian syllogisms from an alternative point of view.

```
"All S are P", i.e. S is included in P
HYPOTHESIS Aps : (x:D) (is-S x) → (is-P x).
```

Note: if HYPOTHESIS **Asp** : $(x:D)(is-P\ x) \rightarrow (is-S\ x)$. is assumed, then S and P are equal.

“No S is P”, and “no P is S”, i.e. S and P are disjoint

HYPOTHESIS **Eps** : $(x:D) (is-S\ x) \rightarrow \sim(is-P\ x)$.

HYPOTHESIS **Esp** : $(x:D) (is-P\ x) \rightarrow \sim(is-S\ x)$.

“Some S is P”, “some P is S”, i.e. non-empty intersection of S and P.

Let $y : D$ be such that $(is-S\ y)$ is the case and let $z : D$ be such that $(is-P\ z)$ is the case.

HYPOTHESIS **Ips** : $(is-S\ y) \rightarrow (is-P\ y)$.

HYPOTHESIS **Isp** : $(is-P\ z) \rightarrow (is-S\ z)$.

“Not all S are P, i.e. some S is not P”

HYPOTHESIS **Ops** : $(is-S\ y) \rightarrow \sim(is-P\ y)$.

HYPOTHESIS **Osp** : $(is-P\ z) \rightarrow \sim(is-S\ z)$.

The predications P and S can be replaced by concrete examples, such as A (for animal),

F (for females), G (for Greeks), H (for humans), S (for singers).

CONSTANTS $is-A, is-F, is-G, is-H : D \rightarrow Prop$.

Some examples of *correct* conclusions.

(**Ahg** d) : $(is-G\ d) \rightarrow (is-H\ d)$ and **Ihg** : $(is-G\ d) \rightarrow (is-H\ d)$.

(**Ahg** d) and **Ihg** have the same type.

Conclusion:

– “Socrates (d) is a Greek, all Greeks are human, so Socrates is human”.

(**Eag** d) : $(is-G\ d) \rightarrow \sim(is-A\ d)$ and **Oag** : $(is-G\ d) \rightarrow \sim(is-A\ d)$.

(**Eag** d) and **Oag** have the same type.

Conclusion:

– “Socrates (d) is a Greek, all Greeks are not animal, so Socrates is not animal”.

Ifs : $(is-S\ z) \rightarrow (is-F\ z)$. some z is F

- “Some singers are female”

Conclusion:

- “Some females are singers”, because z is a female and a singer.

Isf : $(is-Fy) \rightarrow (is-S y)$. some y is S

- “Some females are singers”.

Conclusion:

- “Some singers are females”, because y is a singer and a female.

Ofs : $(is-S z) \rightarrow \sim(is-F z)$. some z is not F

- “Some singers are not female”, “Not all singers are female”.

Osf : $(is-F y) \rightarrow \sim(is-S y)$. some y is not S

- “Some females are not singers”, “Not all females are singers”.

Conclusion:

- Some singers (S) are not female (F), so some females (F) are not singers (S).
Correct, choosing $d : D$ for y and z , because d represents:
 either a singer who is not a female, or a female who is not a singer!

Concluding Remarks

In the present treatise we have explained in somewhat detail the structure of LINCOS and ways of using the system. Evidently LINCOS is useful in describing logic structures in the world as experienced by humans, *e.g.* causal relations in a wide sense, as in the examples listed in the last section. Remarkable in this respect is the simplicity of the system: there are only four basic logic operators ($\wedge$, $\vee$, $\sim$ and $\rightarrow$), there is linear notation, there is the aspect of verifiability within the system—to name some highlights. In addition there is the possibility of exploiting self-explication. We have not actually constructed a message for ETI in this book, partly because that enterprise lies outside the purposes of the book, partly because we feel that this responsible job would best be in the hands of the “Logician in Charge” (see also the DECLARATION OF PRINCIPLES, added to this book).

It has often been remarked that projects concerned with the development of systems such as LINCOS aimed at interstellar communication are of doubtful significance. One reason would be that communication in real time seems to be out

of the question—in view of the limitation of the speed of transfer of information, the upper bound being the velocity of light. Another reason is that serious SETI projects have been carried out for more than 50 years now and no evidence of the existence of ETI has been uncovered. Of course the search has been restricted to only a small (let us say flattened spherical) galactic volume of space around the Sun. By now hundreds of planets have been discovered orbiting their suns in that part of space, but none of them appear to be Earth-like. These arguments point to the conclusion that our intelligent neighbours, if they exist, are not round the corner. In addition these arguments, in view of known stellar densities in the galactic spiral arm we find ourselves situated, lead to an estimate of a lower bound for the average distance between planet-bound intelligent civilizations, of at least 50 light years. If we send a message into space we cannot expect a reply within about 100 years—unless we find a target at relatively short distance.

In our view the significance of projects of the mentioned kind is derived from an entirely different point of view—that can be introduced following an argument put forward by radio astronomer Ray Morris from Australia at the International Astronautical Congress in Melbourne in 1998 (as far as the author is aware, the paper has not been published). Morris' research revealed that the Sun is a relatively young star from the point of view of star formation in the Galaxy. In the galactic space around the Sun there must be many stars thousands or ten-thousands of years older than the Sun, barely or not visible for us.

Supposing that there are planets orbiting suns of this type at distances of say 100–1,000 light years away from us, supporting intelligent life, we might assume that these aliens are, compared to our species

- technologically far more advanced
- seeking knowledge in the same way as we do,
- have developed a *Lingua Cosmica* system of their own,
- and came to the conclusion that the system should be able to explain itself.

If aliens such as these have decided to send messages out in the Galaxy, in the best case directed towards us, informing about their situation, it would be extremely important for us to decode them. This is because we might gain in this way a glimpse of our own possible future, perhaps of a bright kind, perhaps not. In any case, knowledge of this kind would mean a powerful incentive for us to take well care of our planet!

References

- [1] A. Ollongren *On the validation of Lingua Cosmica*, manuscript *in statu nascendi* (2012)
- [2] A. Ollongren *Definition of Programming Languages by Interpreting Automata*, APIC Studies in Data Processing No. 11, Academic Press (1974)
- [3] D.J. Chalmers *The Conscious Mind. In Search of a Fundamental Theory*, Oxford University Press (1996)

Appendix F Summary in Russian

Астролингвистика

Как наука астролингвистика занимается развитием и применением космических языков для межзвездной радиокommunikации между внеземными разумными существами в Галатике или во Вселенной включая выбор научной дисциплины как “общую основу”, например физику, химию, математику или логику. Во всяком случае этот предмет должен иметь универсальный характер. Кроме того, в астролингвистике нужно выбирать пригодные представления формальных выражений.

Первым поколением этих языков был **Линкос**, *Design of a Language for Cosmic Intercourse, Part I*, [Элзевиър 1960] который детально описал со всеми подробностями профессор Др. Ганс Фройденталь († 1991 г.), из Утрехтского университета в Голландии. См. статью Фройденталь *Линкос—междупланетный язык* в сборнике *Населённый Космос* [Фройденталь 1972]. Там автор написал, цитат стр. 310: “Но что же мы будем передавать? С чего начнем? С математики, конечно....”. План этой книги тоже был представлен в интересной статье Бруно Басси: *Were It Perfect, Would It Work Better? Survey of a Language for Cosmic Intercourse* [Stampa (<http://brunobassi.it/scritti/lincos.html>)].

Значительно позже, в Лейденском университете, тоже в Голландии, голландско-шведский астроном и математик профессор Др. Александр Оллонгрэн, в настоящей книге вновь предложил второе поколение космических языков. Новый **Линкос**, или *Lingua Cosmica*, это *система* радиокommunikации, и основывается на двух принципах. Во первых сообщение для внеземных, разумных существ, должно быть многоуровневым. Короткое сообщение состояло бы из части большого текста натурального языка и пополнено аннотациями на другом уровне.

Второй принцип заключается в том что аннотации (или объяснения) должны содержать два важных общих исходных пунктов: дедукцию и индукцию, которые формулируются в *логике*. Т.е. мы будем передавать логические выражения вместо математических. Таким образом логическое

содержание сообщения для межзвездной радиокommunikации, помещается в абстрактный описательный кадр. Этот кадр, логический *database*, содержит константы и переменные величины, определения, предположения, гипотезы и заключения. Как “общая основа” выбирается формальная математическая логика.

Таким образом намеренное составление языка для межзвездной радиокommunikации между внеземными разумными существами, получает новую перспективу. Новая ориентация для *Lingua Cosmica*, определена в настоящей книге, посвященной памяти доктора Фройденталья. Эта книга междисциплинарная, так как она включает прикладную логику, и универсальные аспекты лингвистики (самая суть астролингвистики). К проекту также относятся вопросы о возможности сообщения между внеземными, разумными существами или артефактами в Галатике.

Новый **Линкос** употребляет конструктивную логику с индукцией— специальная модальность математической логики. Таким образом система *Lingua Cosmica* получил солидный фундамент. У **Линкос** есть сигнатура различная от натуральных языков. Выражающая сила **Линкос** большая, но логические выражения часто очень длинные. Для дополнительных сведений, см. *Astrolinguistics, a Guide for Calling E T* [<http://www.alexanderrollongren.nl>].

[1] Элзевир (Elsevier) (1960), Амстердам

[2] Г. Фройденталь (H. Freudenthal), в сборнике *Населённый Космос*, Издательство <<Наука>> (1972), Москва

[3] Stampa, <http://brunobassi.it/scritti/lincos.html>.

[4] <http://www.alexanderrollongren.nl>

The author expresses thankfulness to Alexander Zaitsev and André Deutz for valuable suggestions resulting in a considerably improved version of the first draft of the above summary.

Appendix G Curriculum Vitae of Alexander Ollongren (* 1928, Sumatra)

- University education in mathematics, Hamiltonian mechanics, physics and astronomy, Leiden University in the Netherlands
- PhD in dynamical astronomy, Astronomical Department, Leiden University (1962)
- Post-doc Research Member in celestial mechanics and Lecturer mathematics at Research Center of Celestial Mechanics, Yale University (1965–1967), USA.

Associate director of the computer centre Leiden University (1967–1969) and Lecturer numerical mathematics and computer science at the Department of Applied Mathematics.

Associate professor of theoretical computer science at the Department of Applied Mathematics, Leiden University (1969–1980).

Full professor theoretical computer science at the Leiden Institute of Advanced Computer Science (1980–1993).

- Visiting research member at the IBM Laboratory in Vienna, Austria (1971).
- Sabbatical Leave: Visiting professor at the Department of Computer Science and Artificial Intelligence, Linköping University, Sweden (1980).

Emeritus professor Leiden University, December (1993), public lecture *Vix Famulis Audenda Parat*.

Member of and Professor at the Leiden University Council “University Level Courses for Eldery Citizens” (1995–2005).

Member of the International Astronomical Union (1962–)

Member of IAU Commission 7, Celestial Mechanics (1967–)

Member of IAU Commission 33, Galactic Dynamics (1962–)

Member of IAU Commission 51, Bioastronomy (1998–)

Member of the European Astronomical Society (1980–)

Member of the Dutch Astronomical Society (1955–)

Member of the SETI Permanent Committee of the International Astronautical Academy (2000–)

Nomination as corresponding member of the International Astronautical Academy (2008)

Signatory of “Invitation to ETI, a communication addressed to any Extra-Terrestrial Intelligence having the capability to read it”, (2002–)

Published research papers in:

- dynamical astronomy and celestial mechanics
- theoretical computer science
- formal languages and automata
- formal linguistics in philosophy
- *Lingua Cosmica*

Actor in the documentary film “Calling E T” by Prosper de Roos, Amsterdam, 2008.

Curriculum Vitae of C Bangs

C Bangs art investigates frontier science combined with symbolist figuration from an ecological feminist point of view. Her work is included in public and private collections as well as in books and journals. Public Collections include the Library of Congress, NASA’s Marshall Spaceflight Center, New York City College of Technology, Pratt Institute, Cornell University, and Pace University. Her art has been included in seven books and two peer-reviewed journal articles, several magazine articles and art catalogues. Merging art and science, she worked as a NASA Faculty Fellow; under a NASA grant she investigated holographic interstellar probe message plaques.

www.cbangs.com

Index

A

Abstraction, 10, 14, 17, 29, 39, 42, 47, 57, 80, 95, 96, 102, 123, 131–135, 138
 Actors, 89, 90, 103, 137
 Adam, 120, 121
 Alice, 9–11, 13, 23, 25–27, 119
 Aliens, 102, 189
 Altruism, 80, 101–108
 Anceaux, 200, 201
 Aniara, 163
 Annotations, 1, 2, 39, 40, 48, 52, 65, 67, 72, 75, 83, 86, 87, 89–91, 95, 99–101, 106, 111, 117, 118, 123, 128, 174
 Application, 2, 5, 12, 17, 19, 20, 24, 34, 40, 42, 45, 47, 53, 96, 104, 123, 125, 133, 140, 155–158, 161, 174
 Arbiter, 121, 137, 140, 147–151, 155–156
 Aristoteles, 17, 49, 53–63, 119, 120, 185
 Arity, 5, 6, 10, 68, 69, 95, 96
 of a function, 5
 Artefacts, 71
 Astro-archeology,
 Astrobiology, 151, 159
 Astropaleontology,
 Astrophysical, 101
 time scales, 118
 Astrophysics, 101
 Astropsychology, 204

B

Balloon, 119, 120
 Bangs, 37
 Bassi, 93
 Behaviour, 70, 101–107, 169
 Binary, 4, 11, 16, 21, 79, 106, 133, 145
 Bioastronomy, 116, 175

Biological, 222, 225, 251, 257

Bitmaps, 85, 109

Booleans, 19–21, 77

Bottom, 37

Bracket convention, 42

C

Calculus

 without types, 218

Calculus of Constructions (CC), 1–7, 9, 23, 39, 40, 53, 57, 68, 74, 102, 105, 109

Calculus of Constructions with Induction (CCI), 2, 7, 9, 23, 74, 102, 105

Carrigan, 205

CC. *See* Calculus of Constructions (CC)

CCI. *See* Calculus of Constructions with Induction (CCI)

Chameleon effect, 225, 248

Champollion, 91

Channel, 102, 137, 139–141, 147, 150, 153–156

Chomsky, 75

 generative grammar, 75

Closed, 115, 132

Combinators, 30–32

Combinatory functions, 29

Common ground, 75, 123

Communication, 3, 29, 65, 67, 75, 77, 85, 86, 101, 102, 106, 123, 128, 131, 153, 154, 156, 161, 164, 179, 186, 189, 193

Communication with extraterrestrials (CETI), 85, 86

Commutative, 14, 48, 165

Complex, 49, 52, 66, 109, 121

Complexity, 52, 121

Comprehensibility principle,
 Computation, 5, 37, 161, 164–166, 170, 171
 Computer, 40, 52, 67, 74, 85, 86, 137, 166
 implementation, 74
 Concept, 2, 5, 9, 11, 17, 20, 29, 31, 32, 39, 41,
 68, 75, 78, 93, 94, 96, 97, 99, 102,
 103, 105, 111, 112, 124, 125, 128,
 129, 131, 137, 139, 147, 148, 151,
 153, 156, 159, 161, 164, 166–169,
 178, 179, 183, 184, 186, 189
 mapping, 5, 11, 78, 97, 102, 103, 125, 139
 Conclusion, 2, 12, 23, 34, 38, 41–43, 50, 51,
 56, 60–62, 69, 71, 74, 75, 97, 99,
 105–107, 150–151, 158–159, 174,
 179, 181, 185, 188, 191
 Concurrent process, 137, 139, 140, 147, 148,
 153, 155, 156
 Configurational, 89
 Conjunction, 9, 11, 12, 47, 50, 62, 68, 89,
 113–114, 121, 129, 192
 Connectives, 4–6, 16–17, 47, 133, 184
 Consistency, 109
 Constructive
 proof, 191, 192
 verification, 48
 Constructor, 79, 98, 104
 Contained, 32, 33, 53, 57, 77, 90, 98, 109, 123,
 132
 Context-free, 142
 generative grammar, 242
 Conventions, 1, 29, 30, 32, 41, 53, 60, 68,
 77, 78, 81, 89, 106, 128, 154,
 164, 165, 174
 Conversion rules, 56, 57
 Cooperating process, 137, 140, 147–151, 159
 Coq, 5, 18, 44, 45, 48, 74, 86, 89
 Corpus, 116
 Correctness, 6, 67, 86, 193
 Cosmic, 1, 3, 23, 29, 32, 41, 67, 75, 87, 93,
 101, 102, 109, 121, 123, 129, 131,
 137, 139, 158, 161, 164, 165, 169,
 181, 183, 188–190
 principle, 101

D

Darwin, 220, 227
 Database, 39, 41, 188, 190
 Decidability, 72
 Declaration, 3–7, 10, 11, 15, 18, 23, 39, 45, 58,
 67, 86, 95, 102, 103, 106, 114, 115,
 119, 121, 124, 125, 132, 166, 167, 184
 Declarator, 4, 95
 Deduction, 4, 6, 128

Delimiter, 1, 6, 7, 95, 131
 Denning, 205
 Denomination, 89
 Dialogue, 78, 83
 Disjunction, 9, 11, 13, 14, 47, 68, 114–115, 192
 Distributive, 48
 Domain, 29, 32, 102, 106
 of discourse, 29, 32
 Don't care, 12
 Double negation, 23, 25–27, 48
 Drake, 117
 Dyson, 206

E

Earth, 75, 77, 102, 116, 118, 123, 131, 137,
 164, 178, 179, 189, 193
 Egoism, 103
 Elementary, 11, 29, 32, 41–46, 69, 70, 83, 90,
 102, 139, 147, 151, 153–155, 158
 Elim, 12–14, 17–21, 25, 31, 43–52, 55, 58–60,
 62, 69–74, 111–114
 Elimination, 12, 18, 31, 43–46, 69, 109, 111
 Elliott, 204
 Embedding, 36, 151
 Empathy, 102
 Empty sequence, 35
 Enrich, 166
 Entity, 4, 5, 9, 11, 13, 15–17, 23–25, 37, 42,
 44, 45, 54, 58, 59, 82, 98, 114, 167,
 181, 184, 191
 Environment, 1, 23, 24, 28, 32, 33, 37, 39, 41,
 49, 51, 52, 67–69, 87, 95, 97–99,
 113–115, 132, 134, 139, 151, 181,
 183, 188, 190–192
 Equality, 7, 9–12, 17–19, 27, 41, 94, 142
 relation, 18, 19
 Eva, 120, 121
 Evolution, 101, 118
 Existence, 9, 11, 17–19, 45–46, 50, 51, 58,
 61–63, 67, 68, 72, 105, 114, 132,
 181, 183–188
 map, 184, 185
 Existential issues, 183, 188
 Expressive power, 78, 86, 128

F

Fairness, 151
 False, 19–21, 66, 72, 94, 96, 97, 102, 184, 190
 Falsity, 34, 38, 42, 203, 211, 212, 214
 Formalism, 3, 9, 21, 44, 50, 53, 85, 86, 95,
 102, 109, 128, 132, 164
 Freudenthal, 93, 94, 96

Functional, 5, 42, 74, 97, 125, 181
 application, 5, 42, 125
 Functionality (characteristic)
 of delimiters, 19

G

Galactic, 75, 118, 123
 Galaxy, 75, 118, 193
 Gamelan, 110, 123–129
 Generative, 126
 grammar, 126
 Genetic
 clock, 220
 code, 220
 Gentle, 2, 3
 introduction, 2, 3
 Globally bound, 30
 Gong, 110, 124, 126–129
 Grammatical, 3, 26, 32, 52, 85, 86, 89, 90, 109
 aspects, 32, 85, 86, 90
 base, 3
 formalities, 3
 rules, 109

H

Habitable zone, 199, 210
 Hamlet, 151, 153–159
 Hierarchy, 5, 79, 89, 184
 Holy Grail, 200

Human

altruism, 101–107
 language, 39, 67, 77, 109
 Hypothesis, 6, 7, 12, 17–20, 24, 25, 27, 28, 34,
 39, 43–46, 48, 50, 51, 54, 55,
 57–63, 68, 71, 80, 81, 86, 88, 90,
 91, 105, 113, 114, 132, 134, 181,
 185–188, 191–193

I

Identity
 clause-
 function, 106
 Implication, 4, 9, 11, 12, 14, 16, 17, 23, 41,
 42, 57–60, 67, 97, 102, 106, 115,
 132–133, 135, 184, 190–193
 Impredicative, 183
 Inconsistency, 23, 113
 Individuality, 56, 120
 Induction, 2, 7, 9–21, 23, 31, 34, 43, 45, 50,
 51, 55, 58, 60, 62, 68, 69, 71, 73,
 74, 102, 105, 129, 185–188

hypothesis, 12, 17, 19, 34, 43, 45, 50, 51,
 55, 58, 60, 62, 68, 71, 185–188

Inductive

definition, 7, 11–14, 19, 21, 29–31, 33, 36,
 37, 44, 47, 55, 59, 61, 68, 81, 82,
 98, 102, 104, 105, 114, 115, 129,
 132, 147, 148
 form, 36, 37, 44, 45, 73, 113, 114
 principle, 115, 151

Information content, 52, 85

Intelligent, 65, 75, 77, 94, 102, 109, 115, 118,
 124, 128, 179, 181, 189, 192, 193
 life, 118

Interpretable, 75, 85, 99, 109

message, 85, 99

Interpretation, 5, 11, 26, 29, 39, 40, 51, 52, 54, 65, 75, 77, 78, 86–91, 94, 95, 97, 102–104, 106, 109–116, 123, 124, 126, 131–133, 135, 140, 149, 158, 164, 166, 174, 181, 183

Interstellar

communication, 3, 29, 65, 75, 77, 101, 102,
 123, 128, 131, 161, 164, 186, 189
 message construction, 1, 52, 84, 139

Intra planetary communication, 226

Introduction rule, 11, 13, 16, 17, 114

Intuitionistic, 93, 96, 97

Invariant, 56, 155

Irrational number, 165

J

Janson, 198
 Janus, 70
 Jersild, 220
 Jupiter, 117, 178
 Justice, 78–83, 163

K

Kepler, 178, 179
 Knowledge, 66, 85, 86, 90, 115, 167, 181,
 183, 186
 of the world, 85, 90, 181, 183, 186

L

Lambda

abstraction, 10, 39, 42, 47, 80, 102, 131
 binding, 30, 42, 184–185, 191
 bound, 10, 30, 47, 185
 calculus, 40, 53, 140
 form, 10, 11, 40, 43, 96, 98, 114
 term, 10

Language

- for expressing, 151
- of the genetic code, 220

Leibniz, 7, 10, 19–21

Lemma, 41, 81, 83, 86, 90, 97, 191

Lewis Carroll, 23, 25

Lingua universalis,

Linguistic

- deep structure, 85
- kind, 1, 9, 53, 75, 85, 94, 101, 118, 123, 128, 131, 137
- means, 39, 75, 85, 115, 124, 137, 139
- patterns, 126
- principle, 115
- signature, 131
- system, 1, 9, 67, 77, 94, 99, 115, 116, 118, 123, 128, 139

LISP, 74

List programming, 74

Local, 10, 12, 14, 19, 20, 37, 43, 47, 57, 59, 69, 74, 80, 95, 106, 107, 132, 133, 135

Logic

- symbolic, 67, 75, 123, 132, 161, 165, 174
- two-valued, 72

Logician, 60, 84

- in charge, 84

Looking glass,

M

Martinson, H., 168

Matching, 37

Mathematical foundation, 174

Matrjoshka, 29, 32–37, 40, 98, 99

Mechanical, 65, 68, 70

Mentalese,

Meta

- expression, 86
 - level, 1, 78, 85, 86, 151
- Modelling, 103, 147, 155, 164
- Modus Ponens, 4, 6, 43, 44, 73
- Modus Tollens, 4, 6, 20, 43, 44, 48, 73
- Moralism, 103–105, 107
- Multi-level, 1, 85, 86, 101, 110, 120, 164

N

Negation, 4, 9, 11, 16, 19, 23, 24, 43, 55, 56, 63, 83, 133

Nested structures, 48

Notion, 4, 7, 10, 18, 37, 77, 80, 81, 95, 97, 99, 103, 118, 120, 121, 131, 137, 140, 145, 148, 165–167, 183, 189, 190

- abstract, 77

Nuiten, 70

O

Obligation, 78, 80, 81, 99, 102, 105

Operational, 77, 98, 112, 125, 154, 181, 189

semantics, 77

Operator, 9, 14, 48, 68, 74, 95, 132, 135, 165, 166

Overloaded, 5, 6, 97, 106, 168, 170, 184

P

Paradox, 18, 45, 55

Parametrized, 7, 9, 11, 13, 17

Partial order, 5, 37, 184

Particular, 1, 5, 10, 28, 37, 44, 45, 53, 54, 56, 63, 74, 77, 82, 84, 99, 101, 106, 120, 124, 126, 134, 137–139, 150, 174, 185, 186

Pattern, 60, 115, 126

- recognition, 4

Peirce, 72, 190–193

Platform, 156–158, 161, 189

Predefined, 79, 131, 132, 134

constants, 79

Predicate, 53, 54, 56, 57, 89, 93, 181–185, 189

Predication, 54, 56, 61, 62, 119, 121, 132, 134, 185

Predicative, 183

Premise, 41, 42, 46, 51, 60, 73, 191, 192

Principle

- of choice, 123, 171
- of design, 29
- of verification, 65, 73–74

Processes

- biochemical, 197, 220
- biological, 222, 225, 251, 257
- concurrent, 137, 139, 140, 143, 147, 148, 153, 155, 156
- cooperating, 137, 140, 147–151, 159
- dynamic, 68, 119, 137, 139
- earthly, 137
- evolutionary, 118
- parallel, 143–145, 148–151, 153–156, 159
- sequential, 137, 140, 142, 143, 147–151, 159
- silent, 141–143, 154, 155
- static, 140

Procreation, 186–188

Procreativity, 186

Program, 52, 65, 86, 150, 151, 153–156, 158

Programware, 67

Projection, 4–6, 13, 113, 114

Propagation, 70

Proper

time, 161, 168–174, 178, 179

Properties, 29, 50, 58, 90, 99, 119, 132, 133,
147, 153, 169, 179
basic, 153

Q

Quantifier, 6

Quantized, 132

R

Range, 102, 106, 107, 135, 147

Rational number, 93–96, 165

Reality, 41, 67, 99, 100, 123, 131, 132, 134,
137, 156, 161

Recursion, 29, 32, 33, 35, 82

Redundancy, 39, 70, 109

Relations, 9, 18, 19, 28, 31–33, 48, 65–68, 81,
86, 90, 98–99, 103–105, 110, 118,
119, 121, 132, 137, 141, 181, 185,
188, 190

Resident, 5, 6, 12, 13, 16, 17, 24, 34, 42,
44–46, 51, 55, 59, 68, 69, 73,
79–81, 89, 90, 102, 103, 106, 114,
124, 127, 190, 191

Resolve, 1, 36, 37, 73, 135

Russell, B., 65, 67, 94

S

Sagan, C., 117, 118

Salzman, L., 118, 121

Scope, 65, 67, 86, 106, 135
rules, 135

Selector, 12–14, 16, 17, 19–21, 30, 31, 33, 44,
59, 79, 81, 82, 98, 104, 107, 140,
148, 154

Self

consciousness-

explanation, 85

explication, 259

expression, 84

interpretation, 87, 106, 111–116, 174

knowledge, 86

reflection, 103

verification, 87

Semantics, 1, 75, 77, 83, 109, 115, 116, 132,
148

Separator, 10

SETI, 117, 118, 131

Set of

primitives, 78

rules, 77

Shakespeare, 153, 156

Signature, 39, 110, 131–135

Singular, 49, 56, 120, 185

Socrates, 49, 56, 66, 78, 79, 81–83

Solar system, 117, 177–179

Sound, 24, 67, 87, 110, 113, 124, 126, 127,
129, 181, 188

Species, 75, 77, 123

symbolic, 75, 123

Specification, 138

Stage, 5, 29, 41, 48–52, 54, 55, 60,
68–74, 83, 86, 143,
154, 181

State, 18, 25, 32, 51, 66, 79, 120, 139, 147,
153, 170, 178, 184, 190

Stellar system, 101

Structure, 1, 26, 29–38, 45, 48, 53,
56, 59, 66–68, 85, 86, 98,
99, 102, 103, 105, 124,
126–129, 177

Structuring, 68

Subject, 53, 54, 56, 57, 61, 62, 66, 78,
89, 103, 120, 121, 132, 134,
154, 158, 179, 185

Subvenience, 99–100

Successor, 33, 34, 165, 183

Suffix, 44

Sun, 117, 118, 177–179

Supervenience, 99

Syllogism, 49, 53, 56, 60

Aristotelian, 49, 53

Symbolism, 65–67, 109, 164

Syntax

abstract, 77

concrete, 77

T

Tachyonic velocity, 210

Tautology, 18, 72, 190

Terminals, 35, 36

Tertium non Datur, 97, 190

Textbook, 116

Time, 74, 75, 91, 93, 101, 113, 118, 125,
137, 139, 140, 142, 145, 147,
149, 151, 153, 158, 161, 166–174,
177–179, 186

Top, 142

Tough, A., 193

Tractatus, 65, 94

Transcription, 106

Transitive, 19, 32, 33, 48, 98

True, 19–21, 72, 78, 81, 94, 96, 97, 99, 102,
104, 190

Truth, in language, 93

Two-level, 1, 86, 94

 approach, 86

Type

 checking, 42

 markers, 20

Typeless, 112

U

Universal

 kind, 56

 language, 93

Universe, 37, 38, 55, 77, 85, 89,

 132, 186, 189

Unparametrized, 134

Unsound, 24

Untyped, 3

V

Vakoch, D., 78, 79, 101, 113, 151, 159, 183

Validation, 87

Vector, 140, 142–145, 147, 149–151,
 153–158

Verbal part, 35

Verifiable, 11, 28, 40, 57, 97, 99, 102, 190, 192

Verification

 machinery, 28, 67–68, 74

W

Well-founded/well-foundedness, 5, 37, 38

Wittgenstein, 15, 18, 25, 28, 40, 52, 65–74, 86,
 94, 96, 112, 188, 190, 191

Written representations, 39

 expressions, 35