Lecture Notes in Artificial Intelligence 7254

Subseries of Lecture Notes in Computer Science

Stephen Cranefield M. Birna van Riemsdijk
Javier Vázquez-Salceda Pablo Noriega (Eds.)

Coordination, Organizations, Institutions, and Norms in Agent System VII

COIN 2011 International Workshops
COIN@AAMAS 2011, Taipei, Taiwan, May 3, 2011
COIN@WI-IAT 2011, Lyon, France, August 22, 2011
Revised Selected Papers

Springer

Volume Editors

Stephen Cranefield
University of Otago, Dunedin 9016, New Zealand
E-mail: scranefield@infoscience.otago.ac.nz

M. Birna van Riemsdijk
Delft University of Technology, 2628 CD Delft, The Netherlands
E-mail: m.b.vanriemsdijk@tudelft.nl

Javier Vázquez-Salceda
Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
E-mail: jvazquez@lsi.upc.edu

Pablo Noriega
IIIA-CSIC, 08193 Barcelona, Spain
E-mail: pablo@iiia.csic.es

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-35544-8 e-ISBN 978-3-642-35545-5
DOI 10.1007/978-3-642-35545-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012953566

CR Subject Classification (1998): I.2, I.2.11, D.2, C.2, H.4, H.5, H.3, K.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

This volume is the seventh in a series that started in 2005, and contains the revised versions of 12 selected papers presented at the two COIN (Coordination, Organizations, Institutions and Norms in Agent Systems) workshops in 2011. The first took place on May 3 in Taipei, at the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), while the second was held on August 22 in Lyon, at the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT).

The papers in this collection have undergone a substantial process of refinement. As in previous editions, each submitted paper was reviewed by three Program Committee members and revised versions of the accepted papers were presented in the workshop sessions. After their presentation, some papers were selected to be part of this volume. These selected papers had to be rewritten to take into account the original reviewers' remarks and the comments sparked by the oral presentations. In the case of the papers selected from the COIN@IAT meeting, rewriting has been substantial since the original submissions were relatively short (following conference policy). All revised papers from the two workshops have undergone a second stage of review before producing the final version that is included in this volume.

In keeping with the aim of the COIN workshops, these collected papers share the basic premise of looking into coordination, organizations, institutions, and norms from a *macro* perspective. In COIN, rather than the individual features of the agents themselves, the interest resides in the collective aspects of interactions, the context where interactions take place or the regulations that affect those interactions. While this basic premise is shared, the papers contained in this volume exhibit a healthy diversity of approaches.

There are three papers whose main concern is the coordination and organization of groups of agents. The first two look into the global effect of local interactions. Inspired by ecological systems, Lurgi and Robertson in "Multi-agent coordination through mutualistic interactions" focus on how coordination and communication features among agents (in large self-organizing populations) may achieve desirable network properties. Similarly, in their paper "Explanation in human–agent teamwork," Harbers et al. explore the effects that the explanation of agent actions has on the performance of teams of agents that cooperate using a paradigmatic scenario of human–robot cooperation. The third paper by Keogh and Sonenberg, "Adaptive coordination in distributed and dynamic agent organizations," is interested in modeling the organization of groups of agents where agents not only adapt their own plans to a collective task but also improvise new plans along the way.

The papers by Jiang et al. and by Tampitsikas et al. both have an organizational perspective. Jiang et al., in "An agent-based inter-organizational collaboration framework: OperA+," focus on features that result from the combination of organizations, and in particular (profiting from the OperA framework) how to model collaborative relationships between agents that are members of separate organizations that come together in a partnership. Tampitsikas et al., in "MANET: A model for first-class electronic institutions," propose a framework (extending the OCeAN proposal and formalized using Event Calculus) to model systems where several electronic institutions are situated in a common environment, and where individual agents are active in more than one of these institutions.

Another group of four papers deals with norm-aware agents that reason within a regulated multi-agent system. Panagiotidi and Vázquez-Salceda describe in "Towards practical normative agents: a framework and an implementation for norm-aware planning" a conceptual framework to model practical normative agents that need to reason about norms when planning how to achieve their goals. Letia and Goron, in "Towards justifying norm compliance," propose including the notion of "justification" in normative multiagent systems and use a justification logic and value-based argumentation to that effect. Balke et al. discuss in "Normative run-time reasoning for institutionally-situated BDI agents" a methodology to produce run-time reasoning components for BDI agents starting from a design-time institutional model. Finally, Cranefield et al. demonstrate in "Modelling and monitoring interdependent expectations" the benefits of having "expectations" as a first-class construct in normative multiagent systems and show how to use a temporal logic with expectations and a model checker to handle problematic cases of nested expectations.

Finally, there is a fourth group of papers whose focus is on the process for norm creation and enforcement. In "Operationalization of the Sanctioning Process in Utilitarian Artificial Societies," Balke and Villatoro look into the processes involved in the life-cycle of normative MAS and, in particular, discuss the process of punishment from two perspectives: as a device for norm compliance, and as a device for norm enforcement. Two papers by Mahmoud et al. also look into metanorms in general and punishment in particular, but in the context of norm emergence. While in "Overcoming omniscience for norm emergence in Axelrod's metanorm model" they focus on the way agents learn how to punish, in "Establishing norms for network topologies" they explore the effects of the topology of communication links among agents.

COIN strives to fulfill its workshop role of stimulating discussion, facilitating convergence and synergy of approaches, and weaving a community. Authors and reviewers were encouraged to contribute to a workshop program that welcomes the presentation of unconventional approaches—perhaps stemming from other disciplines—as well as reports about ongoing work, and testimonials of the application of the ideas of this community. The papers in this collection correspond to that invitation.

In terms of their main contribution, this year's papers may be partitioned in three groups: Five papers aim to explore or demonstrate a novel idea (Cranefield et al., Harbers et al., Letia and Goron, and both papers by Mahmoud et al.). Two papers (Balke and Villatoro, and Lurgi and Robertson) propose a new approach; and five papers (Keogh and Sonenberg, Tampitsikas et al., Panagiotidi and Vázquez-Salceda, Jiang et al., and Balke et al.) present a modeling framework.

The papers reflect a diverse pool of influences: modal logics of different flavors (Panagiotidi and Vázquez-Salceda, Cranefield et al., and Letia and Goron), answer set programming (Balke et al.), event calculus (Tampitsikas et al.), argumentation theory (Letia and Goron), normative programming languages (Cranefield et al., Panagiotidi and Vázquez-Salceda, Balke et al., and Lurgi and Robertson), planning (Panagiotidi and Vázquez-Salceda), learning (Mahmoud et al.), complex systems and networks (Lurgi and Robertson, and Mahmoud et al.), experimental economics (Mahmoud et al.), legal theory (Balke and Villatoro), ecology (Lurgi and Robertson), management science (Harbers et al., and Keogh and Sonenberg), and software engineering and sociotechnical systems design (Tampitsikas et al., Keogh and Sonenberg, Jiang et al., Balke et al., and Harbers et al.).

Surprisingly, all but two papers that appear in this collection claim to be motivated, to some extent, by practical considerations. Perhaps future COIN workshops may have papers that analyze the type of applications, problem domains, examples, and illustrative scenarios that are used in this community. Those that are used or explicitly mentioned in this volume are: health care, peer-to-peer computing (file sharing, wireless sensor networks, and wireless grids), traffic management, robotic search, the blocks world for teams, auctions, and running a collaborative project.

Three papers provide methodological guidelines (Balke et al., Jiang et al., and Keogh and Sonenberg). Four papers (Lurgi and Robertson, Balke et al., and both papers by Mahmoud et al.) use some sort of experimental validation or agent-based simulation to back their results, and all but one refer to some sort of implementation of the ideas presented in the paper.

We would like to end this brief preface with two notes. One of gratitude, the other of sorrow. We four, first as workshop chairs and then as editors of this volume, want to express our sincere gratitude to the reviewers of the COIN 2011 editions. Everyone knows that reviewing is not an easy task: it demands generosity—to allocate time and energy that is taken away from other duties; good sense and optimism—to provide constructive criticism; plus a balanced use of confidence, altruism, and courage—to recommend the acceptance or rejection of papers. The names of this year's program committee members are listed for everyone to see in the front matter of this volume, but their contribution is subtly present in the many suggestions that were taken up by the authors to enrich the final version of their papers. Sadly, though, one of these PC members is no longer with us. Marc Esteva passed away last December. He had been a very active member of the COIN community. He wrote a thesis on electronic institutions

and was the engineer behind the EIDE platform for their specification and implementation. At the time of his unexpected death he was directing two PhD students working on COIN topics (immersive regulated environments and the evolution of situated and service-rich electronic institutions), and was the head researcher in a Spanish project in this field. We will remember him in COIN not just because he had been a reviewer in most of the editions and a contributing author to many, but also because he was an enthusiastic participant in the workshop debates. To those of us who had the privilege of knowing him more closely, recalling his many personal qualities will somehow compensate for the weight of his loss.

July 2012

Stephen Cranefield
M. Birna van Riemsdijk
Javier Vázquez-Salceda
Pablo Noriega

Organization

Program Committees

COIN@AAMAS

Alexander Artikis	NCSR "Demokritos", Greece
Guido Boella	University of Torino, Italy
Olivier Boissier	ENS Mines Saint-Etienne, France
Cristiano Castelfranchi	ISTC/CNR, Rome, Italy
Antonio Carlos da Rocha Costa	Universidade Católica de Pelotas, Brazil
Virginia Dignum	Delft University of Technology, The Netherlands
Marc Esteva	IIIA-CSIC, Spain
Nicoletta Fornara	University of Lugano, Switzerland
Jomi Fred Hübner	Federal University of Santa Catarina, Brazil
Catholijn Jonker	Delft University of Technology, The Netherlands
Christian Lemaitre	Universidad Autonoma Metropolitana, Mexico
Victor Lesser	University of Massachusetts Amherst, USA
Eric Matson	Purdue University, USA
John-Jules Meyer	Utrecht University, The Netherlands
Simon Miles	King's College London, UK
Eugenio Oliveira	Universidade do Porto, Portugal
Andrea Omicini	Università di Bologna, Italy
Sascha Ossowski	Universidad Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Alessandro Ricci	Università di Bologna, Italy
Juan Antonio Rodríguez-Aguilar	IIIA-CSIC, Spain
Tony Savarimuthu	University of Otago, New Zealand
Christophe Sibertin-Blanc	IRIT, France
Jaime S. Sichman	Universidade de São Paulo, Brazil, Brazil
Viviane Torres da Silva	Universidade Federal Fluminente, Brazil
Munindar Singh	North Carolina State University, USA
Catherine Tessier	ONERA, France
Leon van der Torre	University of Luxembourg, Luxembourg
Wamberto Vasconcelos	University of Aberdeen, UK
Javier Vázquez-Salceda	Universitat Politècnica de Catalunya – BarcelonaTech, Spain
Harko Verhagen	Stockholm University, Sweden
Marina de Vos	University of Bath, UK
George Vouros	University of the Aegean, Greece
Pınar Yolum	Boğaziçi University, Turkey

COIN@WI-IAT

Huib Aldewereld	Delft University of Technology, The Netherlands
Sergio Alvarez-Napagao	Universitat Politècnica de Catalunya – BarcelonaTech, Spain
Alexander Artikis	NCSR "Demokritos", Greece
Guido Boella	University of Torino, Italy
Cristiano Castelfranchi	ISTC/CNR, Rome, Italy
Antonio Carlos da Rocha Costa	Universidade Católica de Pelotas, Brazil
Stephen Cranefield	University of Otago, New Zealand
Virginia Dignum	Delft University of Technology, The Netherlands
Marc Esteva	IIIA-CSIC, Spain
Nicoletta Fornara	University of Lugano, Switzerland
Jomi Fred Hübner	Federal University of Santa Catarina, Brazil
Christian Lemaitre	Universidad Autonoma Metropolitana, Mexico
Victor Lesser	University of Massachusetts Amherst, USA
Eric Matson	Purdue University, USA
John-Jules Meyer	Utrecht University, The Netherlands
Simon Miles	King's College London, UK
Pablo Noriega	IIIA-CSIC, Spain
Eugenio Oliveira	Universidade do Porto, Portugal
Andrea Omicini	Università di Bologna, Italy
Sascha Ossowski	Universidad Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Jeremy Pitt	Imperial College London, UK
Alessandro Ricci	Università di Bologna, Italy
Juan Antonio Rodríguez-Aguilar	IIIA-CSIC, Spain
Christophe Sibertin-Blanc	IRIT, France
Jaime S. Sichman	Universidade de São Paulo, Brazil
Catherine Tessier	ONERA, France
Leon van der Torre	University of Luxembourg, Luxembourg
Wamberto Vasconcelos	University of Aberdeen, UK
Harko Verhagen	Stockholm University, Sweden
Marina de Vos	University of Bath, UK
George Vouros	University of the Aegean, Greece
Pınar Yolum	Boğaziçi University, Turkey

Additional Reviewers

Sara Casare	Universidade de São Paulo, Brazil
Luciano Coutinho	Universidade Federal do Maranhão, Brazil
Amineh Ghorbani	Delft University of Technology, The Netherlands

Ramón Hermoso	Universidad Rey Juan Carlos, Spain
Özgür Kafalı	Boğaziçi University, Turkey
Henrique Lopes Cardoso	Universidade do Porto, Portugal
Alan Perotti	University of Torino, Italy
Matteo Vasirani	Universidad Rey Juan Carlos, Spain

Workshop Organizers

COIN@AAMAS

Stephen Cranefield	University of Otago, Dunedin, New Zealand scranefield@infoscience.otago.ac.nz
Pablo Noriega	IIIA-CSIC, Bellaterra, Catalonia, Spain pablo@iiia.csic.es

COIN@WI-IAT

M. Birna van Riemsdijk	Delft University of Technology, Delft, The Netherlands m.b.vanriemsdijk@tudelft.nl
Javier Vázquez-Salceda	Universitat Politècnica de Catalunya – BarcelonaTech, Barcelona, Spain jvazquez@lsi.upc.edu

COIN Steering Committee

Alexander Artikis	NCSR "Demokritos", Greece
Eric Matson	Purdue University, USA
Nicoletta Fornara	University of Lugano, Switzerland
George Vouros	University of the Aegean, Greece
Jeremy Pitt	Imperial College London, UK
Julian Padget	University of Bath, UK
Javier Vázquez-Salceda	Universitat Politècnica de Catalunya – BarcelonaTech, Spain
Viviane Torres da Silva	Universidade Federal Fluminente, Brazil
Wamberto Vasconcelos	University of Aberdeen, UK

Acknowledgements

We are grateful to all the conference organizers who accepted our proposals for COIN workshops and the workshop organizers who together created the fora in which our discussions flourish. We are equally pleased to acknowledge the continuing support of Springer, and Alfred Hofmann in particular, for the annual publication of the COIN workshop series, providing both a research record and a dissemination channel to reach those researchers not able to attend the meetings in person.

Table of Contents

Agent Coordination

Organisations and Institutions

Norm-Aware Agent Reasoning

Norm Creation and Enforcement

Multi-agent Coordination through Mutualistic Interactions

Miguel Lurgi and David Robertson

School of Informatics, University of Edinburgh
10 Crichton Street. EH8 9AB, Edinburgh, UK
miguel.lurgi@ed.ac.uk,
dr@inf.ed.ac.uk

Abstract. In this paper we present an ecologically-inspired approach to agent coordination. Mutualistic networks of interacting species in nature possess characteristics that provide the systems they represent with features of stability, minimised competition, and increased biodiversity. We take inspiration from some of the ecological mechanisms that operate at the interaction level in mutualistic interactions, and which are believed to be responsible for the emergence of these system level patterns, in order to promote this structural organisation in networks of interacting agents, enhancing in this way their cooperative abilities. We demonstrate that given plausible starting conditions, we can expect mutualistic features to appear in self-organising agent systems, and we compare them with natural ones to show how the characteristics displayed by ecologically inspired networks of agents are similar to those found in natural communities. We argue that the presence of these patterns in agent interaction networks confer these systems with properties similar to those found in mutualistic communities found in the real world.

Keywords: agents coordination, ecologically-inspired interactions, complex systems, mutualism, emergent behaviour, cooperation.

1 Introduction

Complex systems of interacting entities are ubiquitous in nature and the artificial world. They range from networks of interacting computers in cyberspace to different types of transportation networks (e.g. airports and the links connecting them), power grids that provide cities with electricity, or human interactions of different kinds (e.g. researchers and their collaborations). These highly organised webs of interactions are also found in biological systems such as protein, gene, cell, neural, and ecological networks. The latter kind depicts the relationships found in real communities in a given ecosystem, where species form complex but organised collections of interacting entities.

In societies of artificial agents we are sometimes interested in promoting long-lasting and complex relationships of the kinds described above in order to accomplish difficult tasks that are achieved more easily in a cooperative way.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 1–20, 2012.

In nature, the process of evolution by natural selection has produced different types of interactions between organisms composing communities. Together, these interactions comprise the mechanisms ultimately responsible for the formation and maintenance of the entangled web of life found in ecosystems.

From the range of interactions found in natural communities, we are mainly interested in mutualisms: interactions in which both of the species involved benefit from participating on them. This kind of relationship allows for the creation of relations amongst species that facilitate each others survival. It has been argued that this kind of relation can enhance the stability of the community the interacting species belong to [4] and promote biodiversity within it [5].

In this paper, we explore the extent to which relationships of this kind amongst agents in artificial systems can promote the stability and persistence displayed by communities of interacting species within natural ecosystems.

Ecologically inspired approaches have been applied to the design and development of multi-agent systems in order to promote features such as self-organisation and adaptation of agents in open, digital environments. Some of them have even taken inspiration from the way interactions occur among species in nature; as in [19], where the authors propose a "food-web" based approximation. In this work we go beyond this kind of approaches by deepening in the level of detail for the definition of interactions among agents, based on ecological mechanisms believed to account for important system-level properties.

We describe an approximation for the specification of protocols of interactions for enabling relationships amongst artificial agents living in a multi-agent system based on ecological concepts. We show how the organisation of the interactions among these agents is similar in many ways to the patterns encountered when analysing ecological networks in general and mutualistic webs in particular.

We argue that these specifications at the interaction level will enable our intelligent systems with the features encountered in their natural counterparts, and which undermine the adaptability, persistence, robustness, and stability found in those complex natural networks of individuals.

2 Mutualistic Networks

A focus of research in ecology is based on the characterisation of different kinds of interactions that are observed amongst individuals within ecological communities. The study of the patterns and structure of these interactions and their implications for community organisation and persistence has been tackled through the application of concepts borrowed from the mathematical fields of graph and network theory, giving rise to the ecological discipline of ecological networks.

Studies of this kind performed on natural communities have shown that these systems are generally characterised by small world patterns [12], contributing in this way to fast propagation of the information across the networks; truncated power law degree distributions [11], a feature characteristic of scale-free networks, with a small number of nodes possessing degrees greatly exceeding the average

(usually referenced as *"hubs"*) that are believed to be the strength and, at the same time, the weakness of this kind of network; and the importance of weak interactions for the maintenance of the overall community [6].

A particular type of ecological network, in which the interactions depicted represent mutualistic relationships among the species in the community, are mutualistic networks. Research on this type of ecological webs has demonstrated that they share some of the patterns and features described above with other kind of ecological networks [4], for example their heterogeneity or scale-free character. At the same time however, they are different in many respects, exhibiting properties that are specific to these kind of networks: with specialist species interacting with proper subsets of the species with which more generalist species interact [4], and being pervasively asymmetric in the links between species [17].

The features displayed by mutualistic networks of interactions, and their architecture, are believed to facilitate biodiversity maintenance and to minimise competition in this kind of system [5]. These features are desirable in agent systems in which we would like to promote organisational abilities.

It has been demonstrated in real ecosystems, and particularly in mutualistic communities, that biological and ecological processes describing the interactions between species within them are ultimately responsible for the structure and patterns that we observe in these mutualistic networks of relationships [18].

We are particularly interested in those mechanisms because in this work we take inspiration from some of the processes characteristic of mutualistic interactions, such as: *trait matching*, *habitat* occupation (i.e. spatial distribution), or the formation of *meta-communities*; to define protocols of interaction among agents in agent systems. We employ these notions of mutualistic interactions in the formalisation of protocols of interaction between agents in our agent based system.

The specification of agents' interactions in this way have allowed us to fulfil the objective of promoting collaborative links among agents and the emergence of system level properties that facilitate the system's stability and persistence. We want our societies of artificial agents to display some of the features we encounter in ecological networks of self-organised, autonomous individuals.

Thus, the way mutualistic species interact in nature gives us useful insights about the way artificial intelligent entities may interact in a cooperative digital environment in order to form organised and structured societies.

3 Methods

Since the focus of interest in this research is the design of interactions among agents in a multi-agent system, we adopted an interaction centred approach for the implementation of the system on which to develop the ideas presented in this paper.

We used the OpenKnowledge system [8] as a framework for deploying our multi-agent system. This allowed us to adopt a protocol definition approach for

```
01 a(visitor, X) ::
02  null ← chooseHost(Y, ListOfHosts)
03  then
04  whereabout ⇒ a(host, Y)
05  then
06   (in(HabitatHost) ⇐ a(host, Y)
07    then
08     ((((null ← sameHabitat(HabitatHost)
09        then
10        whichtrait ⇒ a(host, Y))
11        or
12       (null ← metaCommunity()
13        then
14        whichtrait ⇒ a(host, Y)))
15     then
16     availabletrait(Trait) ⇐ a(host, Y)
17     then
18     null ← haveTrait(Trait)
19     then
20     whichsize ⇒ a(host, Y)
21     then
22     size(TraitSize) ⇐ a(host, Y)
23     then
24     (null ← complementary(TraitSize)
25      then
26      null ← need(Amount, Reward)
27      then
28      exchange(Amount, Reward) ⇒ a(host, Y)
29      then
30      offer(Offered) ⇐ a(host, Y)
31      then
32      null ← consume(Offered, Reward)))
33      or
34   (quit ⇒ a(host, Y))))
35 then
36 a(visitor, X)
```

Where *null* denotes an event which does not involve message passing; the operator *::* is used to declare the definition of a role within the protocol; and the operators ←, *then* and *or* are connectives for logical implication, sequence and choice respectively. $M \Rightarrow A$ denotes that a message, M, is sent out to agent A. $M \Leftarrow A$ denotes that a message, M, from agent A is received.

Fig. 1. The "visitor" role in an ecologically inspired interaction model, written in LCC, for multi-agent coordination

```
37 a(host, Y) ::
38  whereabout ⇐ a(visitor, X)
39  then
40   (null ← location(Habitat)
41    then
42    in(Habitat) ⇒ a(visitor, X)
43    then
44    ((quit ⇐ a(visitor, X))
45     or
46     (whichtrait ⇐ a(visitor, X)
47      then
48      null ← myTrait(Trait)
49      then
50      availabletrait(Trait) ⇒ a(visitor, X)
51      then
52       ((quit ⇐ a(visitor, X))
53        or
54        (whichsize ⇐ a(visitor, X)
55         then
56         null ← traitSize(TraitSize)
57         then
58         size(TraitSize) ⇒ a(visitor, X)
59         then
60         ((quit ⇐ a(visitor, X))
61          or
62          (exchange(Amount, Reward) ⇐ a(visitor, X)
63           then
64           null ← obtained(Reward)
65           then
66           null ← has(Amount, Offered)
67           then
68           offer(Offered) ⇒ a(visitor, X)
69           then
70           null ← synthesis(Reward, Growth)
71           then
72           null ← expendResource(Offered, Growth))))))
73  then
74  a(host, Y)
```

Fig. 2. The "host" role in an ecologically inspired interaction model, written in LCC, for multi-agent coordination

the formalisation and description of the interactions among the entities sharing our digital environment. OpenKnowledge makes use of the Lightweight Coordination Calculus (LCC) [14] for the specification of interactions among autonomous agents which communicate through a peer-to-peer (P2P) network.

For describing interactions among agents in multi-agent systems we focus on mechanisms believed to account for system-level patterns in natural communities [18]. Based on key concepts in ecology such as habitat, meta-communities, traits, and resources, we focus in the mechanism of *"trait matching"* and take inspiration from how species in nature are bound together via complementary phenotypic traits to propose a method for describing interactions between artificial agents within a multi-agent system.

Protocols for agent interaction coordination in LCC are based on the definition of roles, which are in turn composed of clauses specifying the actions to be performed by the agents taking part in a particular interaction, when they embark upon any of its instances, by assuming one of the roles available in it. LCC follows a logic programming paradigm, therefore variables are instantiated through constraint calls. Agents' roles defined in this way are thus composed of a series of constraints that are executed and verified in order to instantiate the variables declared along the protocol. These variables encapsulate the information to be transmitted within the messages that are exchanged in-between constraints (see figures 1 and 2).

We defined the roles of "visitor" and "host" inside our protocol of interaction in resemblance to the actors taking part in plant-animal mutualistic interactions in nature. Figures 1 and 2 show the definition of the interaction protocol, comprising the definitions of the roles mentioned above. In this protocol, it is clear how the notions from mutualistic interactions introduced above have been formalised as an agent coordination protocol. This also implies that the implementation of agents in the system is driven by these concepts, since agents must contain the functions, and have to be able to handle the messages, contained in the protocol. In sum, these ideas are thus the main drivers for the design of the multi-agent system itself. The protocol formalisation is further explained below, where the general idea of the interaction and a description of the concepts involved in it are presented.

Agents implementing the roles mentioned interact through the exchange of a series of messages generated via local constraint solving. Through these messages they exchange and process information about: (i) the habitat they occupy within the environment, (ii) the size of the trait through which they are interacting, (iii) their ability to form a meta-community, and (iv) the amount of resource they are going to exchange if the interaction is successfully completed. All of these events can be seen in our interaction protocol shown in figures 1 and 2.

Information about the size of the interaction traits is used to obtain a degree of complementarity, which partially determines the actual occurrence of the interaction. Meta-communities are formed when agents interact with partners located in habitats different than their own.

The messages to be exchanged and the constraints to be solved during the execution of the interaction protocol possess a certain ecological meaning and have been formulated in order to specify the ecological process of interaction as described above. The exact meaning of each of them is the following:

- *chooseHost*: through the resolution of this constraint, the agent taking the visitor role selects a partner for interaction among the possible host agents available in the system. This is a random selection.
- *whereabout*: the visitor agent sends out this message to the host partner in order to obtain information about the location (i.e. the habitat) of the host.
- *location(Habitat)*: when asked about the habitat it occupies, the host agent employs the 'location' constraint to determine it in order to be able to share this information.
- *in(HabitatHost)*: the host agent replies to the visitor's request sending this message, which contains the value of the habitat where the host is located (HabitatHost).
- *sameHabitat(HabitatHost)*: using the information of the habitat of the host (just obtained from the previous message exchange) the visitor agent uses this constraint to analyse this value and determine whether the host agent lives in the same habitat as it does.
- *whichtrait*: following the interaction protocol, if the previous constraint yields true, the visitor agent will employ the *whichtrait* message to ask the host for the trait to be employed during this interaction.
- *metaCommunity*: if the sameHabitat constraint returns false, the visitor agents employs the metaCommunity constraint to determine whether it is able to establish a meta-community with the host agent, even though they are in different habitats, if this is not the case, then the visitor agent sends out a 'quit' message to the host and the interaction is aborted.
- *availableTrait(Trait)*: if the interaction continues, the host agent will reply to the visitor request with the availableTrait message, which will be used to inform the visitor which trait the host is willing to employ during this interaction.
- *haveTrait(Trait)*: the visitor agent is able to evaluate the response of the host using the haveTrait constraint, which will take the Trait value received from the host and determine whether the visitor can cope with that trait. If that is not the case, then the agent will send a 'quit' message to the host to inform that the interaction has been broken.
- *whichsize*: this message is employed by the visitor agent to ask the host the size of the trait being employed. In real life, trait sizes correspond to the magnitude used to quantify the given trait, e.g. the size of the bill of a hummingbird in respect to the size of the corolla of the flower it will pollinate. This size, as it happens in nature, will be used by the agent to determine whether it is complementary with the host.
- *size(TraitSize)*: the request from the visitor agent is answered by the host through this message, in which it sends the information about the size of the trait that was selected for the current interaction.
- *complementary(TraitSize)*: through this constraint, the visitor agent is able to determine whether the traits selected for interaction are complementary among the partners. If that is not the case, the visitor agent ends the interaction by sending the host the 'quit' message, otherwise the interaction continues.

- *need(Amount, Reward)*: if the interaction can continue, the visitor agent utilises this constraint to determine the amount of resource it is expecting (i.e. it needs) from the interaction and the reward it is willing to offer to the host agent.
- *exchange(Amount, Reward)*: the information obtained through the previous constraint is shared with the host agent in order to find out whether it is able to provide the requested resource and is happy with the offered reward. This is done using the 'exchange' message.
- *obtained(Reward)*: using this constraint the host agent considers whether the reward offered by the visitor agent is enough for the interaction to take place.
- *has(Amount, Offered)*: based on the amount of resource requested by the visitor, the host agent calculates the amount that it is actually capable to offer and will employ this information to inform the visitor about it.
- *offer(Offered)*: once the amount of resource to be offered has been calculated by the host agent, it sends this information to the visitor using this message.
- *consume(Offered, Reward)*: the visitor agent employs this constraint to determine whether the offered resource by the host is enough to cover its needs, or simply if it is willing to accept it. If this is the case, it will consume the resource and update its internal state accordingly based on the resource consumed and the reward offered to the host. If it is not willing to take the offer, it will send out the 'quit' message to the host to let it know that the interaction was cancelled.
- *synthesis(Reward, Growth)*: if the visitor accepts the offer, then the host agent will use this constraint to synthesise some resource based on the reward obtained from the visitor and will produce some growth resource.
- *expendResource(Offered, Growth)*: the growth resource along with the amount of resource offered to the visitor will be considered using this constraint to calculate the net gain of the host for this interaction.

Both roles in the protocol are recursive, as defined by the last line of each role, which resets the execution of the protocol and allows the agents to embark on future interactions. It is also worth noting the strategic locations of the 'quit' message, which can be used at any time to abort the interaction when certain conditions necessary for it to happen are not met (e.g. agents in different habitats or with not complementary trait sizes).

As shown by the protocol, this technique for agent communication relies heavily on the complementarity of traits of the partner agents for any given interaction, which means that agents aggregate in groups in which each agent is able to provide different functions according to the necessities of the other partners in the community. It is key for this approach then, that agents are able to perform different roles in a given association or conglomerate, in order to ensure the stability of the network by minimising competition and increasing diversity.

This ideas are in sharp contrast with other approaches in which agents aggregate based on the affinity of their capabilities and features, like for example [3] and [7], where the mechanisms of local convergence and homophily ensure that agents with similar capabilities tend to develop stronger connections.

Our model has a number of parameters and settings that can be configured in order to produce different outcomes given different circumstances for the agents to interact. Our default general settings however, define certain values that are common to all of the simulation results presented. The main settings are:

1. number of agents: in resemblance to the proportion of plants versus pollinators in plant-animal interaction networks, where there are generally more plant species than animal ones, we set the number of host agents in our agent systems to ten and that of the visitors to fifteen.
2. number of habitats: the habitat a given agent occupies will partially determine whether it will be able to interact with the agent selected for interaction; they will be more likely to interact if in the same habitat than otherwise. In our experiments the default number of habitats employed is two, and agents are evenly distributed among both habitats.
3. minimum and maximum resources: any given interaction is based around the exchange of resources that will ultimately allow agents to survive; at the beginning of each run, agents are given a certain amount of resources that is selected randomly from a normal distribution bounded by a minimum and maximum amount of resources. By default the range for this amount is between five (min resource) and twenty (max resource). Although the initial amount of resource may determine the fate of an agent, these numbers were selected in order to ensure stability in resource distribution.
4. meta-community formation: agents possess the ability of interacting with individuals belonging to different habitats than their own; when this happens they are said to form a meta-community. The probability of any given agent forming a meta-community association is 0.1.
5. maximum trait size: in our model we have a parameter specifying the maximum size of the interaction trait, and all the agents will have sizes ranging from zero to this maximum size. The default value for the maximum size of traits in our system is five. The size of a trait will determine the extent to which partners in an interaction will be complementary to one another, thus, this parameter is a measure of overlap of features (the closer the size of the traits on complementary partners the more likely they will be able to interact).

Sensitivity analyses have been performed on the initial values of the parameters described above in order to ensure that different configurations of these parameters do not produce a noticeable variation on the behaviour of the system (data not shown [10]).

The analyses which follow are based on multiple series of simulations, configured as described above, with initially random encounters between agents. These allow us to generate networks of interactions for analysis.

Our study of the networks obtained in this way takes advantage of a set of metrics borrowed from the field of ecological networks and which are commonly used to analyse this type of complex network of interactions. It is important to note here that some of the measurements that can be obtained from interaction

networks of this kind are calculated on a particular representation of them: the interactions matrix.

The interaction matrix is a matrix where the interactions between species / agents in the network are represented by the cells in it; in the case of a qualitative representation, it is a binary matrix with a 0 representing the absence of a given interaction and 1 its presence. In its quantitative version, the positions of the matrix with values different from 0 record the strength of that interaction. Since ours is a bipartite network, columns and rows represent different sets of agents, whereas in networks such as food webs, for example, the species are represented by rows and by columns. In our graphical representations the host agents are represented by the columns and visitors are represented by rows.

The metrics we used in this work for analysing our interaction networks are the following:

1. connectance ($C = L/S^2$): the connectance of the network gives a quantitative value of the proportion of links that are actually realised (L is the number of links in the network) relative to the total number of possible interaction links in the network (S is the number of nodes).
2. frequency distribution of the number of interactions: represents the distribution of the frequencies with which we encounter a node with a certain number of interactions.
3. degree distribution: the degree of a node is the number of links it has. The distribution of degrees is a good indicative of the extent to which a given network possesses scale-free features.
4. nestedness index: originally inspired by the theory of island biogeography [2], nestedness is a measure of the organisation of the community which assesses the extent to which the diet, or sets of interactions, of more specialist species (i.e. those possessing only one or a few interactions) are proper subsets of more generalist species.
   It is not difficult to see, based on this definition, that more nested communities (i.e. those in which the interactions are organised in proper subsets among species) display a larger cohesion, since the removal of one random species can be compensated for by the other nested interactions.
   Based on the interactions present in the interaction matrix, a way of calculating how nested the interactions are, is through the isocline of perfect nestedness. It is a measure introduced by Rodríguez-Gironés and Santamaría in [15], which by taking into account the number of interactions occurring in the network, obtains an estimate of a perfectly nested matrix and yields a curve that allows us to visualise the pattern that we should expect from it. This curve, plotted on top of the actual interactions matrix gives an idea about how nested our matrix is.
   This measure is used to determine how far from a perfectly nested community (in regards to the interactions between agents) ours is, and facilitates the visualisation of the interactions that are contributing towards its organisation and those that are keeping it away from it.

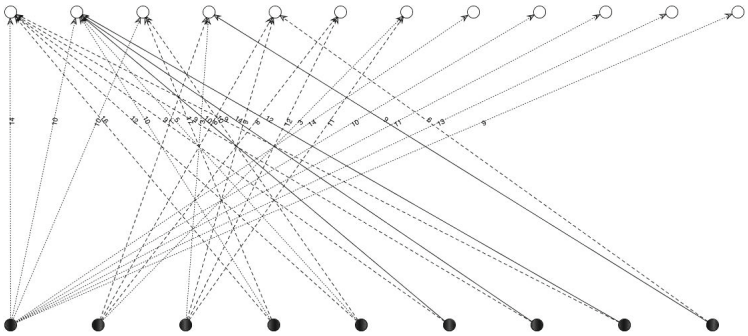


Fig. 3. Network of interactions between agents produced as an output of a simulation run performed in a multi-agent system, using the interaction model described in this paper

Additionally, we calculate the Nestedness metric based on Overlap and Decreasing Fill (NODF), as presented in [1], and which is another metric commonly used in the analysis of nestedness in ecological networks [9] for determining the extent to which a given interaction network presents a nested pattern.

4 Results

Our experiments consisted of the execution of a series of independent simulation runs following the specifications outlined above for parameter initialisation and run configuration. During these runs, relationships among pairs of agents arose with different strengths (the number of times an agent interacts with any other relative to the number of times it has interacted during the entire simulation) and with different configurations.

4.1 Ecologically Inspired Mutualistic Agents Networks

Figure 3 shows an example, taken from one of the runs in our experiments, where the relationships among agents in our multi-agent system are represented in a fashion similar to networks of interacting species described in the field of ecology.

In this graph, the agents and their relationships with other agents are represented by nodes and arcs respectively. A link (arc) is generated between two agents whenever an interaction is successfully completed amongst them.

By representing the relationships between the agents in agent systems in this way we are able to extract features, obtain descriptors, and perform analyses over the resulting network based on methods borrowed from network theory.

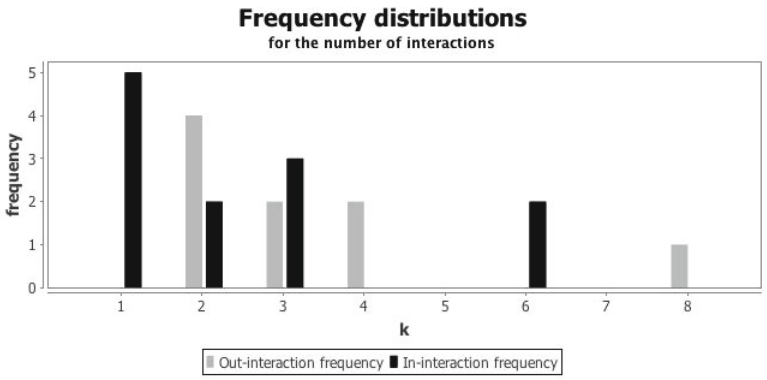


Fig. 4. Frequency distribution of the number of interactions for the network in Fig. 3

We obtained networks that display a scale-free structure: the plot displayed in figure 4 shows data from the network in figure 3, where it can be observed that the majority of nodes in this network have small degree (≤ 2), while a small fraction of them are highly connected, showing a distribution of the frequency of interactions biased towards low values (1 and 2 interactions). Additionally, small-world properties are found in our networks: with short paths between any two nodes, a property commonly found in ecological networks.

Properties of the kind mentioned above, which are encountered in the networks of interactions among our agents, are common patterns also found in different kinds of complex networks in nature and the artificial world [16], and which differ significantly from the structure that we would expect from a randomly assembled network.

Another property seen in our networks, which is related to their scale-free character, is the preferential attachment displayed by visitor agents with low degrees (e.g. the four white nodes on the top right corner of figure 3) to host agents that are highly connected. This is a common feature encountered in mutualistic networks, where specialist species are more likely to interact with generalists [4]. Patterns of this kind are important in practice because, as has been argued, they can give us information about functional properties of the system such as: information propagation speed and resistance to node failures [16], which in turn provide us with a better understanding of the relationship between the complexity and stability of our agent systems.

Particularly, in the realm of ecological networks, asymmetric specialisation (i.e. a specialist interacting with a generalist) has been found to be a pervasive feature of plant-pollinator interaction networks, and it is believed to be beneficial for the majority of species in these communities because it facilitates the avoidance of extinction risks when species are highly reciprocally specialised [17].

Table 1. Connectance and NODF values of the fifteen simulated and fifteen empirical networks analysed in this paper

Sim	C (L/S^2)	NODF	Emp	C (L/S^2)	NODF
1	0.053	25.203	SAPF	0.031	18.36
2	0.062	37.063	CACO	0.034	29.693
3	0.064	32.456	CAFR	0.039	34.166
4	0.066	28.667	SCHM	0.041	56.66
5	0.067	36.132	MOMA	0.045	32.067
6	0.068	33.333	GEN1	0.061	34.243
7	0.07	42.913	OFLO	0.062	35.961
8	0.077	61.06	BAIR	0.064	50.98
9	0.079	44.409	ESKI	0.071	54.586
10	0.079	51.404	BEEH	0.074	67.66
11	0.083	54.82	WYTH	0.075	45.411
12	0.083	70.102	KANT	0.084	67.344
13	0.086	68.528	LOPE	0.103	57.423
14	0.087	59.211	HRAT	0.111	78.756
15	0.087	63.739	FROS	0.163	74.571

The network architecture displayed by the interactions amongst agents is an emergent property of our system since the only mechanisms involved in agents' interactions are those specified by the protocol presented in section 3. The creation of such a complex and intricate pattern of relationships is not a hardwired property of the artificial communities, but rather the product of many different agents interacting together for achieving their respective goals (gathering resources to survive).

In the context of this system, biodiversity refers to the number of agents with different functions, or in our case interactions, that inhabit the digital environment. The fact that agents assuming different roles and possessing different traits and/or sizes are able to live in the system is a clear sign that diversity is enhanced in our system. Even though agents and their parameter values are initialised randomly at the beginning of each run, as explained above, when choosing partners for interactions, only those that comply with the biologically mutualistic notions introduced in section 3 are selected, promoting thus the emergence of the interaction patterns we encounter (e.g. figure 3).

Agents would want to communicate and form complex networks of interactions in order to obtain enough resources to survive in the digital environment. Because of the benefits outlined above, agents that are incorporated into the network will not only ensure their survival, but they will also contribute towards the organisation of the community. This will in turn, through the indirect effects propagated across the network - and ensured by its structural patterns - minimise the competition between agents, further benefiting them as a group and contributing towards their sustainability.

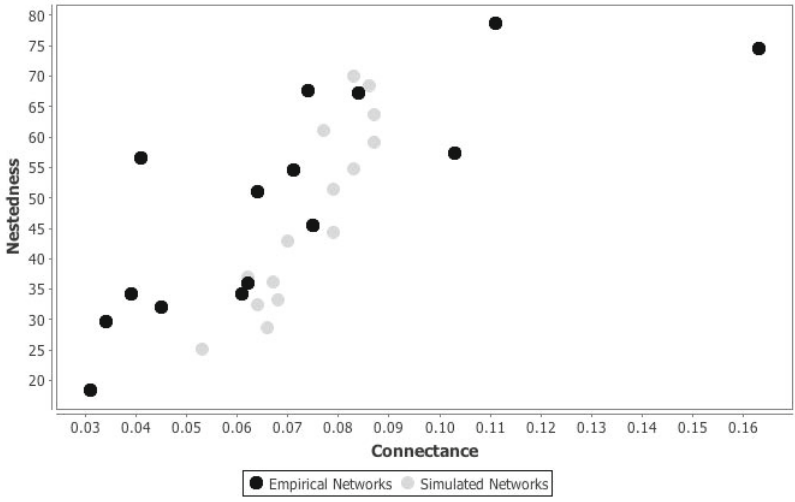


Fig. 5. Connectance versus NODF nestedness values in the natural (black dots) and simulated (grey dots) communities presented in Table 1

4.2 Comparing with Empirical Data

In order to test the extent to which the networks of interactions found in our artificial systems are similar to mutualistic networks of interactions found in the real world (apart from the qualitative similarities introduced above), we have compared the architecture of the networks obtained from our simulations to empirical data of networks collected from real communities and that have been compiled, analysed, and provided as supplementary material by Rezende et al in [13]. Although in that paper they used the networks for different kinds of analyses, the datasets provided are useful for getting an idea of the common features encountered in mutualistic networks.

Because some of the properties of interest for analysing ecological networks are scale dependent, we have selected fifteen networks from this dataset, based on the number of species composing it and that were closest to the number of agents in our simulations, for comparison against our fifteen simulated networks.

As mentioned in section 3, connectance ($C = L/S^2$), the fraction of all possible links that are realised in a network, is an important property that is commonly employed in the analysis of ecological networks and which provides with information about the degree of connectivity between the nodes of the network. We use the connectance and the NODF index of nestedness, introduced in section 3, for comparing our simulated networks with fifteen empirically obtained plant-animal mutualistic networks. Additionally we perform qualitative comparisons between the structures obtained in one of our networks and one of the natural communities considered for this analysis.

In table 1 we can see the connectance and NODF values derived from our simulated networks and the selected empirical networks obtained from natural

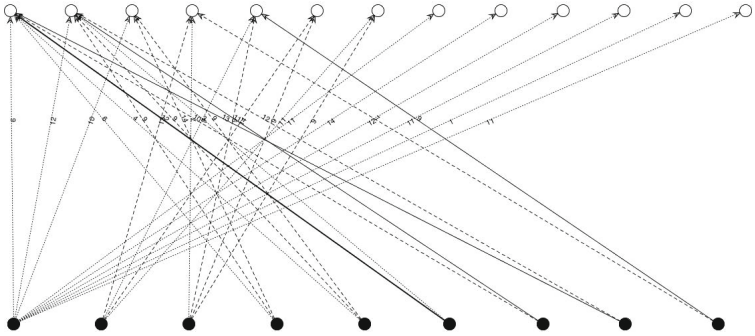


Fig. 6. Network representation of community number 3 presented in Table 1

communities; the names of the empirical networks are as presented in [13], and the numbers used to identify the simulated networks are only for identification purposes and are not related to any feature of the network itself. Figure 5 displays the plot of the NODF against the connectance values with the double purpose of analysing the behaviour of nestedness in relation to changes in the connectance of the network, and to compare these relationships in our simulated communities against their natural counterparts.

As we can see, in both types of networks there is a positive relationship between the connectance of the network and the nestedness value. This is not surprising, since the more connections a network has the more they can contribute to the nested diets observed in those networks. It is perfectly possible however, that more links could mean less nested communities (if they go to the wrong side of the perfect nestedness isocline), so, it is important to bear in mind that in well organised communities, like the ones we consider in this work, the more connected networks are, the more nested they become.

These data (table 1 and figure 5) also show us that the values of connectance and nestedness obtained from our simulated communities agree with the values of these measures commonly found in natural communities, where the connectance is normally between 0.03 and 0.1, and the nestedness values are usually found between 20 and 80. In our communities, the connectance values were greater than 0.05 and lower than 0.1; similarly, the NODF values for our communities were distributed along the 25-80 range of values. This provides more evidence for the self-organised character of our digital societies, a distinctive feature inherited from their biological peers.

4.3 Structural Similarities between Ecological and Agents Networks

We want to further explore the similarities between our agent systems and natural communities; for this we have selected, from the set presented above, one of our simulated and one of the empirically observed systems to perform a one to one comparison.

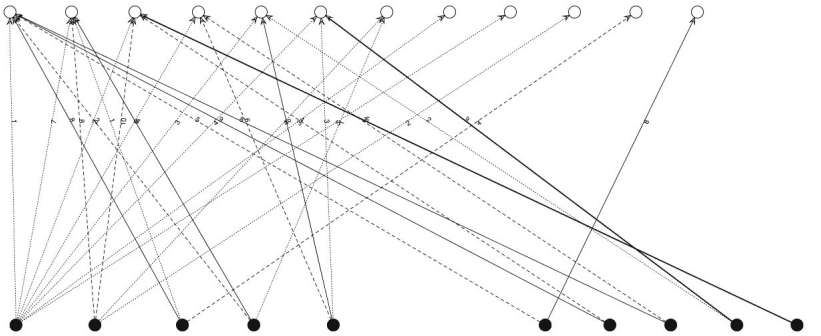


Fig. 7. Network representation of the OFLO natural community presented in Table 1

Two networks were thus selected based on their similarities not only in connectance and nestedness values but also taking into account other features, as we shall see in the following paragraphs. The networks selected were: the one represented by the number 3 in table 1, with values 0.064 and 32.456 of connectance and NODF index for nestedness respectively, for representing our simulated communities; and the OFLO natural community, with 0.062 connectance and 35.961 NODF index, as a representative of the natural communities considered.

Figures 6 and 7 show the network of relationships between entities in the simulated community number 3 and the OFLO natural community, respectively.

When closely inspecting these two networks we can easily observe a number of broad similarities such as the high proportion of visitor nodes (nodes in white) possessing only one interaction and also an important fraction of host nodes (nodes in black) with two interactions or less. Also notable is the presence of one highly connected host node in both of the networks (black node in the bottom left corner in each of the networks), and also on the side of the visitor species/agents (white node on the top left). These nodes act as generalists/hubs, bringing cohesion and reachability to the whole network.

Another important feature captured by looking at the network of interactions and that is shared by our natural and artificial communities is the low fraction of strong dependencies among nodes (solid dark edges in the graph) and the abundance of weak dependencies, which in natural communities is believed to account for the stability and resilience of these systems, since the loss of a link can be easily adjusted for by resorting to other connections in the network. In agent systems, this property can be translated into the ability of agents to quickly adapt to missing links and not to depend strongly in any other agent in the network, facilitating in this way the persistence of each individual agent and the system as a whole.

Apart from the graphical representation of the interactions network, we also analyse certain properties derived from its structure and that can serve for deepen our comparison among the selected networks.

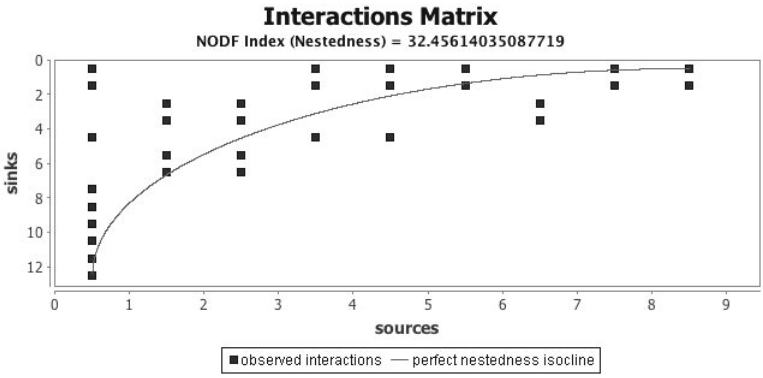


Fig. 8. Matrix of the interactions found in community number 3 in our simulations (including NODF index and perfect nestedness isocline)

Natural communities, as we have seen above, are somewhat less predictable than our simulated ones. This can be confirmed when comparing the interaction matrices of our simulated community number 3 and the OFLO natural community (figures 8 and 9 respectively): community 3 presents a much more organised structure, with few interactions below the isocline of perfect nestedness, while the OFLO community presents not only more interactions below the isocline but also a few of them that are actually far removed from it. Also the degree distributions in both communities, although agreeing in the overall scale-free pattern, differ subtly by the fact that the values in our simulated community are closer to the fitted power law (data not shown).

In spite of these differences, the communities present very similar distributions of the frequencies of the number of interactions (data not shown), with practically all the nodes possessing less than five interactions and the majority of them with two or one (this can be appreciated in the graph representation of the networks). Additionally, only three nodes in the case of community 3 and two in the case of the OFLO community possess more than five interactions. This reaffirms the scale-free character present not only in the natural communities employed here as reference, which is expected from this type of networks, but also in our simulated communities as an emergent feature of the self-organisation in our agent systems.

Our natural and simulated communities are similar not only in terms of the network structure, but also in terms of their features and characteristics. These similarities let us see how mechanisms implemented at the individual interaction level between artificial agents allow for the development of mutualism through agent interaction. Agent interactions defined in this way facilitate thus the emergence of system level properties that are commonly encountered in natural communities of mutualistic species. As has been argued throughout this paper, these features can provide our self-organising agent societies with stability and robustness.

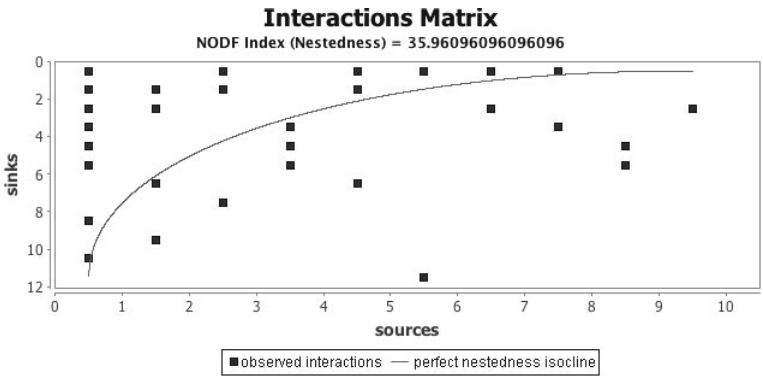


Fig. 9. Matrix of the interactions found in the OFLO natural community (including NODF index and perfect nestedness isocline)

By comparing the structure of the networks of relationships resulting from our simulations, with the networks of interactions amongst mutualistic species in nature, we can translate the results of the research performed until now in the field of ecology in general, and ecological networks in particular into the domain of multi-agent systems. The characteristics described above, which are shared between our networks of artificial agents and those observed between species in the natural world, let us conclude that the features conferred to these natural systems by the characteristics displayed by their networks of interactions are also given to our agent systems thanks to the emergence of a similar network structure to that found in these ecological networks.

5 Conclusion

The ecologically inspired techniques for agent coordination and communication presented in this work allow societies of autonomous agents to form self-organised complex networks of interactions that benefit from many of the features encountered in ecological networks of interactions among mutualistic partners in nature.

Since the systems from which we have taken inspiration are composed of mutulistic relationships arising in nature, and by equating the patterns encountered in the network of interactions between agents in our simulated communities to those seen in their natural counterparts, our systems also benefit from the properties of increased biodiversity and minimised competition characteristic of these type of natural systems [5], which promotes organisation and collaboration between the artificial entities in our intelligent systems.

To our knowledge no attempts have been made so far to develop agent interactions inspired directly by ecological theory, which we think is a fruitful research area. Furthermore, in the field of multi-agent systems, little effort has been devoted to the analysis of this kind of system from the point of view of complex systems. The tools provided by the field of complex networks in general and

ecological networks in particular, and that we have used as part of our study, constitute then, as far as we know, one of the first attempts to analyse complex networks of agents' interactions in multi-agent systems from a complex systems perspective.

Acknowledgments. Special thanks are due to Michael Rovatsos for presenting this work during the COIN workshop at AAMAS 2011 on behalf of the authors. Also, we would like to thank Michael, Stephen Cranefield, and an anonymous reviewer for useful comments that helped us to improve this manuscript towards its present form.

This work is part of the project "EcoBusiness: A Multi-Agent Approach for Digital Business Ecosystems", funded by the European Commission through the Seventh Framework programme Marie Curie Actions - Industry-Academia Partnerships and Pathways (IAPP). Grant agreement no.: 230618.

References

1. Almeida-Neto, M., Guimarães, P., Guimarães Jr., P.R., Loyola, R.D., Ulrich, W.: A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement. Oikos 117(8), 1227–1239 (2008)
2. Atmar, W., Patterson, B.D.: The measure of order and disorder in the distribution of species in fragmented habitat. Oecologia 96(3), 373–382 (1993)
3. Axelrod, R.: The dissemination of culture: A model with local convergence and global polarization. Journal of Conflict Resolution 41(2), 203 (1997)
4. Bascompte, J., Jordano, P.: Plant-animal mutualistic networks: the architecture of biodiversity. Annu. Rev. Ecol. Evol. Syst. 38, 567–593 (2007)
5. Bastolla, U., Fortuna, M.A., Pascual-Garcia, A., Ferrera, A., Luque, B., Bascompte, J.: The architecture of mutualistic networks minimizes competition and increases biodiversity. Nature 458(7241), 1018–1020 (2009)
6. Berlow, E.L.: Strong effects of weak interactions in ecological communities. Nature 398(6725), 330–334 (1999)
7. Centola, D., González-Avella, J., Eguíluz, V., Miguel, M.S.: Homophily, cultural drift, and the co-evolution of cultural groups. Journal of Conflict Resolution 51(6), 905 (2007)
8. de Pinnick Bas, A.P., Dupplaw, D., Kotoulas, S., Siebes, R.: The openknowledge kernel. International Journal of Applied Mathematics and Computer Sciences 4(3), 162–167 (2007)
9. Joppa, L.N., Montoya, J.M., Solé, R.V., Sanderson, J., Pimm, S.L.: On nestedness in ecological networks. Evolutionary Ecology Research 12(1), 35–46 (2010)
10. Lurgi, M.: Ecologically inspired agent networks. Master's thesis, University of Edinburgh. College of Science and Engineering. School of Informatics (2010)
11. Montoya, J.M., Pimm, S.L., Solé, R.V.: Ecological networks and their fragility. Nature 442(7100), 259–264 (2006)
12. Montoya, J.M., Solé, R.V.: Small world patterns in food webs. Journal of Theoretical Biology 214(3), 405–412 (2002)
13. Rezende, E.L., Lavabre, J.E., Guimarães Jr., P.R., Jordano, P., Bascompte, J.: Non-random coextinctions in phylogenetically structured mutualistic networks. Nature 448(7156), 925–928 (2007)

14. Robertson, D.: A Lightweight Coordination Calculus for Agent Systems. In: Leite, J., Omicini, A., Torroni, P., Yolum, P. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 183–197. Springer, Heidelberg (2005)
15. Rodríguez-Gironés, M.A., Santamaría, L.: A new algorithm to calculate the nestedness temperature of presence–absence matrices. Journal of Biogeography 33(5), 924–935 (2006)
16. Strogatz, S.H.: Exploring complex networks. Nature 410(6825), 268–276 (2001)
17. Vázquez, D.P., Aizen, M.A.: Asymmetric specialization: a pervasive feature of plant-pollinator interactions. Ecology 85(5), 1251–1257 (2004)
18. Vázquez, D.P., Blüthgen, N., Cagnolo, L., Chacoff, N.P.: Uniting pattern and process in plant-animal mutualistic networks: a review. Annals of Botany 103(9), 1445–1457 (2009)
19. Villalba, C., Zambonelli, F.: An nature-inspired approach for large-scale pervasive service ecosystems. In: Proceedings of the 4th AAMAS Workshop on Massive Multiagent Systems. IFAAMAS (May 2009)

Explanation in Human-Agent Teamwork

Maaike Harbers[1], Jeffrey M. Bradshaw[2], Matthew Johnson[2], Paul Feltovich[2], Karel van den Bosch[3], and John-Jules Meyer[4]

[1] TU Delft, P.O. Box 5031, 2600 GA, Delft, The Netherlands
m.harbers@tudelft.nl
[2] IHMC, 40 South Alcaniz, Pensacola, FL 32502, United States
{jbradshaw,mjohnson,pfeltovich}@ihmc.us
[3] TNO, P.O. Box 23, 3769 ZG, Soesterberg, The Netherlands
karel.vandenbosch@tno.nl
[4] Utrecht University, P.O. Box 80.089, 3508 TB, The Netherlands
j.j.c.meyer@uu.nl

Abstract. There are several applications in which humans and agents jointly perform a task. If the task involves interdependence among the team members, coordination is required to achieve good team performance. This paper discusses the role of explanation in coordination in human-agent teams. Explanations about agent behavior for humans can improve coordination in human-agent teams for two reasons. First, with more knowledge about an agent's actions and plans, humans can more easily adapt their own behavior to that of the agent. Second, with more insight in the reasons behind an agent's behavior, humans will have more trust in the agents, and therefore more easily coordinate their actions. The paper also presents a study in the BW4T testbed that examines the effects of agents explaining their behavior on human-agent team performance. The results of this study show that explanations about agent behavior do not always lead to better team performance, but they do impact the user experience in a positive way.

1 Introduction

When the members of a team jointly perform a task, it often happens that one team member is dependent on other team members for achieving a subtask. For instance, it may happen that a team member can only start to achieve subtask A after someone else has achieved subtask B, or that a team member can only start to achieve subtask C after another team member has started to achieve subtask D. When team members are dependent on each other for achieving a task, they are *interdependent* [16]. It is inefficient when two interdependent team members are separately trying to achieve a task that can easily be achieved by one team member. Therefore, interdependent team members need to coordinate their actions in order to achieve a good team performance. The better the actions of different team members are coordinated, the higher the performance of the team will be.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 21–37, 2012.

Coordination of actions is not only important in human teams, but also in teams that consist of a mix of humans and software agents. Therefore, when developing agents that aim to participate in a human-agent team, it is important to make them able to coordinate their actions with other team members (both humans and agents). Johnson et al [16] stress the importance of taking the interdependence of the team task into account when designing agents that are to function in a human-agent team. They promote a teamwork-centered approach when designing autonomous systems, called *coactive design*.

In this paper we will analyze literature on teamwork and explanation, and argue that explanation plays an important role in achieving good human-agent team performance. Namely, in order to coordinate actions, it is important that team members understand and predict each other's behavior, and explanations can help to improve insight in other team members' behavior. Consequently, we argue that to develop agents that perform well in human-agent teams, the agents should be equipped with explanation capabilities.

Furthermore, we will describe an empirical study investigating the role of explanation in human-agent teamwork. For the study, we will use the BlocksWorld for Teams (BW4T) testbed for team coordination [18]. In BW4T, a team of players has to perform a joint task in a virtual environment. The players are highly interdependent, and the performance of the team strongly depends on the level of coordination among the players.

The outline of this paper is as follows. In section 2, we will discuss literature on teamwork and explanation, and motivate why explanation is important in human-agent teamwork. In section 3, we will describe the BW4T coordination testbed for investigating teamwork. In section 4, we describe the experiment that examines the effect of explanations about agent behavior on the coordination in human-agent teams. In section 5, we end the paper with a conclusion.

2 Background

In this section we will discuss human teamwork, human-agent teamwork, and explanation in human-agent teams, respectively. Human teamwork has been studied for several decades. Compared to the large body of literature concerning human teamwork, there is relatively little literature on human-agent teamwork. The work on human-agent teamwork builds on concepts and theories that were developed in research on human teamwork. Therefore, before we discuss human-agent teamwork specifically, we first provide a short introduction to human teamwork.

2.1 Human Teamwork

There are two main streams in the literature on human teamwork. In the first, the concept of transactive memory is used to explain teamwork, and in the second, the concept of shared mental models is used to explain teamwork. We will describe both views.

Transactive Memory Systems. The theory of transactive memory was first introduced by Wegner [35]. A transactive memory system (TMS) is a memory system that is distributed across different team members. In a TMS, each of the team members has 1) knowledge that captures his or her own expertise, and 2) knowledge about who knows what. The knowledge that needs to be remembered is thus divided over the different team members. The assumption is that it is more efficient for an individual to remember who has knowledge on a certain topic than remembering all the details by oneself.

In order to use TMS theory for the explanation and prediction of team performance, different ways to measure TMS have been proposed [24,1,27]. Moreland et al [24], for example, distinguished three components of TSM: specialization, credibility and coordination. The specialization component refers to the level of knowledge differentiation within the team. Credibility refers to team members' beliefs about the accuracy of other members' knowledge. Coordination refers to team members' ability to work together efficiently.

Results of TMS as a determinant of performance are promising [1,21,25]. However, a real consensus among researchers on how to measure TMS is lacking. First, there is no commonly accepted theory on which components comprise TSM. Second, there are different ways to measure a team's performance on these components.

Shared Mental Models. Mental models refer to the internal representations that humans have of the world around them. Mental models enable humans to understand, explain and predict the systems in their environment [28]. In the context of teamwork, mental models can help individuals to understand the behavior of other team members and to predict their future actions. This allows the individuals to adjust their own actions to the expected behavior of others. It is argued that in order to coordinate the actions of different team members well, it is important that the team members have similar mental models: shared mental models (SMM) [6]. Most researchers classify SMM into two broad dimensions: task-related knowledge and team-related knowledge (e.g. [7]). Task-related knowledge concerns knowledge about how to achieve the task, the current status of task achievement, etc. Team-related knowledge concerns knowledge and capabilities of other team members, what they are currently intending or doing, etc. Experimental results trying to demonstrate the effects of sharedness of mental models on team performance are promising. However, like for TMS, there is no common method for measuring the sharedness of mental models [23].

Relation Between TMS and SMM. There is little interaction between the research fields of TMS and SMM. An exception is the work of Nandkeolyar [25], who compared both theories on their predictive power on team learning and team effectiveness. He found that in most cases high levels of TMS components (specialization, coordination and credibility) and high levels SMM both predicted team performance well. However, in some cases, high levels of SMM did

not result in high team performance, especially when teams scored high on TMS specialization and credibility.

Researchers from both sides have stated that one theory is an extension of the other. Shared knowledge in SMM theory can be seen as a team member's knowledge about who knows what in TMS theory. The other way around, a team member's knowledge about who knows what in TMS theory can be seen as shared knowledge in SMM theory. Whether both theories only provide a different vocabulary for the same processes, or describe distinctive phenomena, in both cases do the two theories have a different focus. TMS focuses more on the dividedness of knowledge and SMM focuses more on the sharedness of knowledge. Both sides, however, do acknowledge that some of both is needed. Without any shared knowledge, it is not possible to coordinate actions, but totally overlapping knowledge leads to a single minded view on tasks, also called *groupthink* [14].

2.2 Human-Agent Teamwork

Literature shows that sharedness and dividedness of knowledge are both important in human teamwork. In this section, we argue that sharedness and dividedness of knowledge are at least as important in human-agent teamwork.

Dividedness of knowledge is particularly important in human-agent teams because agents and humans have different strengths and capabilities. For example, agents may be better at remembering a large amount of data than humans, but humans are often better at recognizing danger than agents. Both humans and agents even have capabilities that the other does not have. On the one hand, there is no human that can calculate as fast as an agent can, but on the other hand, there are no agents that can break the ice (socially). To fully benefit of the strengths and capabilities that the members of a human-agent team offer, the tasks should be divided over the team members in such a way that each team member performs the tasks that best suit his or her capabilities and knowledge. For most tasks, especially the complex ones, this will lead to a division of knowledge over team members.

Sharedness of knowledge is important in human-agent teams to coordinate actions, especially because knowledge and capabilities are often divided over team members. When the members in a team have different strengths, they must be are aware of each others' specialties in order to allocate subtasks to the right team member. Moreover, initially humans know less about the behavior of an agent team member than the behavior of a human team member. Namely, being a human already reveals many properties of a team member, e.g. memory capacity, speed of doing tasks. Among agent team members, there is more diversity concerning these properties. Therefore, it is especially important that mental models about what team members know and can do are shared for the coordination of actions. In line with this argument, several approaches for team agents have been proposed that are explicitly based on SMM theory [19,37].

In literature on human-machine interaction, there is a shift of attention from dividedness towards sharedness in human-agent teams. In the seventies, Sheridan

and Verplank [29] introduced different levels of autonomy. At the highest autonomy level, the computer decides everything, acts autonomously, and ignores the human, and at the lowest autonomy level, the computer offers no assistance, and the human must take all decisions and actions. This model of autonomy levels thus focuses on how tasks are divided over machines and humans. Johnson and colleagues [17] argue that the levels autonomy model falls short on the actual complexity of effective human-agent teamwork. They observe that humans and agents have different capabilities and argue that to combine their strengths, it is crucial to have good coordination in human-agent teams [3]. To coordinate actions it is necessary to exchange information about each others' goals, intentions, and observations. This stresses the importance of sharedness of knowledge.

We believe that explanation can contribute to coordination in human-agent teams in two ways. First, explanations about agent behavior can increase the sharedness of mental models by informing humans about the actions, observations and intentions of the agents. With this knowledge, humans will be able to better understand and predict new agent behavior, which will make it easier to coordinate actions. Second, explanations about agent behavior can increase humans' trust in agents. Members of a human-agent team usually have different knowledge and capabilities. So when a team member provides information other team members, e.g. informing about an intention, they will only use that information to coordinate their actions when they trust the team member. Having insight in another's reasoning increases trust, and in human-agents teams trust will improve coordination. In the next section, we will provide a short overview of research of explaining intelligent systems.

2.3 Explanation in Human-Agent Teams

To discuss different applications in which intelligent agent and/or system behavior is explained, we will use Sycara and Lewis' [31] distinction of different roles of software agents in human-agent teams. According to them, agents in a human-agent team can have the role of individual assistant, team assistant and equal team member. We will discuss the explanation of intelligent system behavior for each of these roles.

In the first role, an agent provides individual assistance to a human. In that case, the agent cooperates with only one human, who may or may not be part of a bigger team. Examples of providing explanations as an individual assistant are expert systems and recommender systems. These types of systems both support a single human user in making decisions. The explanation of intelligent system behavior was first researched in the field expert systems. It was discovered that to accept an advice or diagnose of an expert system, users want to know how and why a certain outcome was reached [30,36,11]. Aims of explanations in expert systems are increasing user acceptance, trust, ease of use, usefulness and user satisfaction [10]. Aims of of explanations in recommender systems are transparency, scrutability, trust, effectiveness, persuasiveness, efficiency, satisfaction [32].

In the second role, an agent provides team assistance. A team assistant agent cooperates with all team members, usually to support coordination activities in

the team [33]. The concept of team assistance is relatively young, and we are not aware of explanation approaches for team assistant agents.

In the third role, an agent acts as a (more or less) equal team member. In this role, an agent performs the reasoning and tasks of a human teammate. Virtual training is a field in which the behavior of agents in the role of an equal team member is explained. In virtual training, intelligent agents are used to play a trainee's colleagues, opponents or team members. Several approaches for explaining the behavior of such agent have been proposed [15,34,8,12]. Explanations in virtual training aim to increase the trainee's understanding of the played session, and thereby support learning.

Different roles of agents in teams yield different types of explanations. Expert system behavior (where the system has the role of personal assistant), for example, is explained by traces of rules that were applied and the justification behind those rules [11]. Behavior of agents in virtual training (where the agent has the role of equal team member) is explained in terms of goals and intentions [8,12]. The difference between expert systems on the one hand, and virtual intelligent agents on the other hand, is that the behavior of the latter more closely resembles human behavior. Humans explain and understand their own and others' behavior in terms of the underlying mental concepts such as desires, plans, beliefs and intentions [22,20]. In Dennett's words [9], people adopt the *intentional stance* towards virtual intelligent agents, i.e. they attribute beliefs and goals to them in order to understand their behavior. Thus, the role of the agent in a human-agent team should be taken into account when developing its explanation capabilities.

In the remainder of this paper we will discuss how to study the effects of explanation in human-agent teamwork. The BlocksWorld for Teams coordination testbed provides a mean to investigate human-agent teamwork. We will first describe the testbed itself, and subsequently, a study we performed in the testbed. Agents in BlocksWorld for Teams have the role of an equal team member.

3 The BW4T Coordination Testbed

BlocksWorld for Teams (BW4T) is a testbed for team coordination [18]. In the BW4T testbed, teams of humans, agents, or humans and agents can perform a task that requires coordination in a controlled environment. We therefore believe that the BW4T testbed is a useful tool for studying teamwork. The task is simple to learn and it is possible to manipulate all conditions in the environment, but at the same time, there are many interdependencies among the different players and complex process arise. In this section we will describe the BW4T task, discuss the behavior of a BW4T agent, and discuss the implementation of a BW4T agent.

3.1 The BW4T Team Task

The BW4T task can be performed by human-human, agent-agent and human-agent teams of variable sizes. The team goal is to jointly deliver a sequence of

colored blocks in a particular order as fast as possible. A complicating factor is that the players (human or agent) cannot see each other. Figure 3.1 displays a screenshot of a BW4T game session, showing the environment in which the players have to search for blocks. The left picture displays all blocks and players in the game, and the right picture shows what one player can see. A player can only see the blocks in a room when he is inside that room. The status bar below the Dropzone (gray area) shows which blocks need to be delivered.

Fig. 1. Simulator view with two agents (left) and agent view with one agent (right). The blocks that need to be delivered are respectively orange, light green, dark purple, light blue, dark green and red. Bot0 in the right hall is holding an orange block and bot1 in room C2 is holding a light green block.

To deliver a block successfully, a player has to go to a block of the right color, pick it up and drop it in the Dropzone. A player can only carry one block at a time. When a player drops a block of the wrong color in the Dropzone or any block in a hall, the block disappears from the game. Human players can perform actions in the environment through a menu that appears on a right mouse-button click. The menu offers options to go to a place (room, hall or Dropzone), pick up a block, drop a block and send messages.

A team's performance on the BW4T task is measured by the speed of completing the task. BW4T is designed such that the task involves a large amount of interdependence among the players, and requires coordination to achieve a good performance. For instance, it is inefficient when one player is searching in a room that has just been checked by another. And if a player is going to deliver a particular block, the others should not do that as well. To coordinate, players can send messages to each other, which appear in the chatbox below the Dropzone. Players can inform others about what they do, where they are and what they see. Furthermore, players can see the same status bar. So when a player delivers a block of the right color, the other players will know. Finally, only one player can be inside a room or the Dropzone at the same time. When a player tries to enter a room that is occupied, a red bar appears indicating that someone is inside.

3.2 Behavior of a BW4T Agent

Developing an agent that can perform the BW4T task on its own is rather straightforward. The agent needs to be able to search for blocks and deliver blocks, and it has to plan its behavior. Planning involves deciding what to do (search or deliver), where to search for blocks and which block to deliver. There are several strategies to perform the BW4T task. The agent can for instance search all needed blocks and then deliver them. It can also search for the next block in the sequence and deliver it once found, or keep checking rooms on the way to the Dropzone to deliver a block.

The agent's behavior gets more complex when there is a team of players involved. Each of the agent's has to coordinate its behavior with the others to avoid that a room is checked twice, or that two agents are delivering a block of the same color when only one block of that color is needed. To coordinate, the players have to update others about their activities and percepts, e.g. tell others what they are going to do and which blocks they found in which rooms. Moreover, they have to adapt their own behavior to messages they receive from others. For example, if a red block needs to be delivered and another player says it is going to deliver that block, it is better to search for the next block in the sequence.

When the behavior of the other players in a team is known, it is sufficient to send updates and process updates from others for effective task performance. However, in applications with human-agent teams, usually the behavior of the others is not completely known. The behavior of the agents may be designed by different developers, and behavior of human players can never be completely predicted as humans tend to vary their strategy, make mistakes and forget things. It may happen, for instance, that a player tells that there is a yellow block in room C1, but once you arrive it is not there, or that a player announces that he is going to deliver an orange block, but actually does not, or that someone delivered a white block, even though you had told to deliver it. Therefore, a BW4T agent should be able to deal with unexpected events.

3.3 Implementation of a BW4T Agent

Currently, there are two ways to implement a BW4T agent. The first way is to use Java. BW4T is implemented in Java and offers a basic agent class in which the behavior of a BW4T agent can be specified. The second way is to use `GOAL` [13], a BDI-based (Belief Desire Intention) programming language.

BDI-based programming languages offer the possibility to represent an agent's behavior in terms of its beliefs and goals, and a BDI agent's actions are determined by a deliberation process on its beliefs and goals. The BDI-based agent programming paradigm is based on Bratman's theory of human practical reasoning, in which human reasoning is described with the notions of belief, desire and intention [5]. Rao and Georgeff developed a BDI-based software model [26] based on Bratman's theory. A typical BDI deliberation cycle contains the following steps: (i) perceive the world and update the agent's internal beliefs and goals accordingly, (ii) select applicable plans based on the current goals and beliefs, and add them to the intention stack, (iii) select an intention, and (iv) perform the intention if it is an atomic action, or select a new plan if it is a subgoal.

Currently, there is a set of BDI-based agent programming languages [2] and `GOAL` is one of them. A connection has been established between BW4T and `GOAL`, which makes it possible to implement BW4T agents in `GOAL`.

4 Experiment

In this section we describe the experiment performed in BW4T. As motivated in section 2, we believe that human-agent teams in which agents explain their behavior coordinate better than human-agent teams in which agents do not explain their behavior. In the experiment, we will use performance on the BW4T task to measure the level of coordination in human-agent teams. Our hypothesis is that human-agent teams in which agents explain their behavior perform better on the BW4T task than human-agent teams in which agents do not explain their behavior.

4.1 Method

Design. The experiment has a within-subjects design with an explanation and a no-explanation condition. In the explanation condition, the subjects cooperate with an agent explaining its behavior, and in the no-explanation condition, subjects cooperate with an agent that does not explain its behavior. The order of the two conditions, explanation and no-explanation were assigned counterbalanced to the subjects, to correct for possible learning effects from the first to the second trial.

Subjects. A total of 16 subjects (male = 14, female = 2) with an average age of 27 (sd=3.5) participated in the experiments.

Materials. We used the BW4T testbed described in section 3. In order to investigate the effects of an agent's explanation on human-agent team performance, we developed a BW4T agent that is able to explain its behavior. We implemented the agent in GOAL.

The agent's behavior is formed by the following rules. The agent starts to check rooms and once it knows about a block that can be delivered, it starts to deliver that block. The agent uses information about blocks in rooms received from the other player. When the other player announces that he is going to check a particular room, the agent will not check that room. When the other player tells that he is going to deliver a block, the agent will start to search or deliver the next block in the sequence. The agent is able to deal with humans that vary their strategy, make mistakes and forget to tell things. Namely, the agent revises its plans when a room contains other blocks than it expected, and when the agent holds a block that is not needed anymore, it will drop the block in a room. Thus, in general, the agent is cooperative and assumes that the other player is cooperative as well.

The following GOAL code shows a part of the agent's planning behavior in which it decides to either deliver a block or check a room.

```
IF a-goal(deliverSequence), bel(me(Me),available(Me),
   toPickUp(Block,Color),in(Block,Room))
THEN adopt(delivered(Block)) + insert(delivering(Me,Block)).

IF a-goal(deliverSequence), bel(me(Me),available(Me),
   not(toPickUp(Block,Color)),nextRoomInSeq(Room),
   not(checked(Room)),not(checking(_,Room)))
THEN adopt(checked(Room)) + insert(checking(Me,Room)).
```

The first if-then rule states that if it is the agent's goal to deliver the sequence of blocks, and it believes that it is available to do something and that there is a block that can be picked up, then it will adopt the goal to deliver that block and obtains the belief that it is delivering that block. The second if-then rule states that if it is the agent's goal to deliver the sequence of blocks, and it believes that it is available to do something, there is no block that can be picked up, and the next room that has not been checked is not already being checked by someone else, then the agent will adopt the goal to check that room and obtains the belief that it is checking that room.

As the aim of this study is to study the effects of explanations about agent behavior on coordination in human-agent teams, the agent needs to be able to explain its behavior. In section 2, we argued that agents that play the role of an equal team member are considered intentional. In other words, we understand their behavior by attributing beliefs, goals and intentions to them. We therefore believe that the beliefs, goals and intentions underlying the agent's actions comprise useful explanations about its behavior. The implementation in GOAL allowed us to explain the agent's behavior in terms of beliefs, goals and intentions [12].

To explore the effect of explaining agent behavior on coordination in human-agent teams, we need to be able to manipulate the agent's communication behavior. Inspired on the KaOS policy framework [4], we use policies to regulate the agent's communication behavior, so we do not have to change the agent's programming code. We distinguish the following three communication policies.

1. Inform other players about your observations
2. Inform other players about your actions
3. Provide explanations for your actions

The first policy entails that if the agent observes something in the virtual environment, it sends a message to inform all other players about its observation. Such messages are, for example, 'Room A1 contains a pink block and a dark blue block' and 'Room B2 is empty'. The second policy prescribes that if the agent performs an action, it has to send a message to inform all other players about it. Messages informing about actions are for instance 'Im going to Room C1', 'I picked up a red block' and 'I just dropped a gray block'. The third policy prescribes the agent to explain an action, that is, to provide the underlying goal of that action. In the next section we will discuss the explanation of actions in more detail. Examples explanations for actions are 'I am going to Room B3 to search for an orange block' and 'I am going to Room C2 to deliver a light green block'.

In the explanation condition, the agent adhered to all three communication policies, and in the no-explanation condition, only communication policies 1 and 2 were applied. Thus, the agent equally often provided updates in both conditions, but the updates in the explanation condition were longer than those in the no-explanation condition.

Measures. Team performance was measured by the time of completing the task. Faster task completion indicates a higher team performance. Additionally, the subjects' estimation of team performance, their understanding of the agent's behavior, and their opinion on the length of the explanations was measured by a questionnaire.

Procedure. The subjects received an explanation of the BW4T task and how to direct their 'bot'. Subsequently, they had to play a training session, in which they had to deliver three blocks on their own. The training session was included to make sure that the subjects completely understood the game, and to give them time to think about their strategy in the actual trials. No agent participated in the training session yet, to prevent that it would shape the subjects' expectations about the agents in the trial sessions.

For the two trial sessions, subjects were instructed to perform the task with the agent as a team, as fast as possible. They were told that the agent could show any kind of behavior, e.g. not search in the right places or not take the subject's messages into account, but that the agent would not lie to them. In both trial sessions, the human-agent team delivered six blocks of different colors. The colors and positions of the blocks differed per session, but the total traveling distance

to deliver all blocks was the same. The order of the two conditions, explanation and no-explanation were assigned counter-balanced to the subjects, to correct for possible learning effects from the first to the second trial. After both sessions, the subjects were asked to fill in a short questionnaire.

4.2 Results

The time of completing the BW4T task was used as a measure for team performance. In the explanation condition, the average time (n=16) to complete the task was 596 seconds (sd=118), and in the no-explanation condition the average time was 593 seconds (sd=81). These averages are obviously not significant (paired t-test: p=0.95).

We also examined if there was a learning effect between the first and second session. The average time (n=16) to complete the sessions was 617 seconds (sd=118) for the first session, and 572 seconds (sd=76) for the second session. The results show that the subjects completed the task faster in the second session than in the first session, but the difference is not significant (paired t-test: p=0.26).

In the questionnaire administered after each session, we asked subjects to judge their own, the agent's and their common performance on a scale from 1 to 7. Table 1 shows the averages in both the explanation condition (EX) and the no-explanation condition (NE).

Table 1. Average estimation of performance on a 1-7 scale (n=16)

	EX	NE
I was effectively performing the task	5.9 (sd=0.7)	5.8 (sd=1.1)
The agent was effectively performing the task	6.0 (sd=1.3)	5.5 (sd=1.3)
We were effectively performing the task as a team	5.7 (sd=1.6)	5.1 (sd=1.7)

The results are not significant (paired t-tests: p=0.67, p=0.36, p=0.41, respectively), but for all questions and in particular for agent and team performance, the subjects judged performance on average higher in the explanation condition than in the no-explanation condition, even though no actual differences in performance were found.

In order to investigate how well subjects evaluate performance, we calculated the correlations between the self-evaluations in Table 1 and the actual team performances. Surprisingly, the subjects' self-evaluations have a low or even negative correlation with the actual performances. Three of the negative correlations are significant (α=0.05): evaluated human performance and actual team performance in the no-explanation condition (R=-0.49), evaluated agent performance and actual team performance in the explanation condition (R=-0.50), and evaluated team performance and actual team performance in the explanation condition (R=-0.55). The results show that subjects make better estimates of their own performance in the explanation condition, and better estimates of the agent's and the team's performance in the no-explanation condition.

In the questionnaire, we also asked the subjects to judge how well they understood the actions and motivations of the agents, and how well the agents seemed to understand their actions and motivations. The results in Table 2 show that the subjects had a significantly better idea of what the agent was doing in the explanation condition than in the no-explanation condition (paired t-test: p=0.030). Though the other results are not significant, for all questions understanding was on average rated higher in the explanation than in the no-explanation condition (paired t-test: p=0.74, p=0.65, p=0.47, respectively).

Table 2. Average understanding of behavior on a 1-7 scale (n=16)

	EX	NE
I had a good idea of what the agent was doing	6.1 (sd=1.0)	5.1 (sd=1.4)
The agent seemed to have a good idea of what I was doing	5.8 (sd=1.1)	5.7 (sd=1.0)
I understood the reasons behind the agent's behavior	5.9 (sd=1.2)	5.7 (sd=1.5)
The agent seemed to understand the reasons behind my behavior	5.6 (sd=1.0)	5.3 (sd=1.9)

Finally, we asked subjects if the agent provided *too little*, *just enough*, or *too much* information. In the explanation condition, 1 subject thought that the agents provided too little information, and all other 15 subjects thought that the agent provided just enough information. A chi-square goodness of fit test shows that the result is significant (χ^2=26.4, p<0.001). In the no-explanation condition, 10 subjects indicated that the agents provided too little information, while 6 subjects indicated that the provided information was just enough. This result is significant as well (χ^2=9.5, p=0.009). Thus, in general subjects preferred the amount of information in the explanation condition over the amount of information in the no-explanation condition.

4.3 Discussion

We found no significant differences between human-agent team performance in the explanation and the no-explanation condition. Therefore, the results do not support our hypothesis that explanations about agent behavior improve human-agent team performance on the BW4T task. The experience of the subjects, however, was affected by the agent's explanations. The subjects' ratings of their idea of what the agent was doing was significantly higher in the explanation condition than in the no-explanation condition. Furthermore, a significant number of subjects believed that the agent in the no-explanation condition provided too little information, whereas a significant number of subjects indicated that the agent in the explanation condition provided just enough information.

With a larger number of subjects, more of the results obtained from the questionnaire may have been significant. Namely, all of the subjects' ratings are higher for the explanation condition than for the no-explanation condition,

both concerning self-evaluations on performance as understanding of each other's actions. It is not probable that the difference in performance on both conditions quickly would have become significant with a larger number of subjects, since the performances on both conditions are rather similar.

There are several possible explanations for the similar team performances on both conditions. We provide five of them. First, subjects may have lost time in processing the agent's explanations, which then was compensated by a more efficient task completion. The robots in BW4T move slowly on purpose to provide players sufficient time to communicate, and think and process information. However, at some points in the game many actions have to be done at once (enter a room, go to a block, pick up a block, go to the Dropzone, and communicate about your actions) despite of the slow speed of the robots. Thus, at those time points, processing explanations may lead to time loss.

Second, the subjects may have anticipated a cooperative agent. Though we told them that the agent could perform any behavior and made them aware of possible strategies, several of the subjects reported that their strategy was to behave as if the agent was cooperative until they would find out otherwise. With such a strategy, explanations do not contribute to a quicker adaptation to the agent's behavior as the subject's initial behavior already makes the right assumptions about the agent's behavior. It would be interesting to conduct an experiment with a less cooperative or capable agent, e.g. one that cannot process certain messages or is colorblind, to see if explanations help subjects to quicker adapt to the gaps in the agent's capabilities.

Third, the task may involve too much noise. Some of the subjects, for instance, reported that they mistook one color for another (e.g. yellow and light green), which caused a serious delay. Other subjects said that they changed their strategy after the first trial, e.g. they let the agent deliver all blocks. Furthermore, though the blocks are evenly spread over the rooms in different trials, there is a luck factor involved in finding blocks. This factor can be decreased by letting the team deliver more blocks, but adding blocks also gives the subjects more time to learn the agent's behavior, which decreases the expected effect of providing explanations. In conclusion, noise factors like these may have wiped out the effects of explanation on team performance.

Fourth, the task may be too simple to show an effect. In most situations, the rationale behind the agent's behavior can be deduced from its actions.

Finally, the agent always explained its actions by the goals they aimed to achieve. The advantage of such explanations is that they are immediately derivable from the mental state of a BDI agent. Possibly, when extending the agent's explanation capabilities, e.g. by adding information about the agent's strategies, the explanations would become more useful and have a bigger effect on team performance.

5 Conclusion

In this paper, we discussed literature on human teamwork, human-agent teamwork and the explanation of intelligent systems and agents. We argued that

explanation of the behavior of agents in a human-agent teams can contribute to team performance in two ways. First, when team members have more shared knowledge, e.g. about their current activities and plans, it is easier to coordinate their actions. Second, explanations can increase trust in a team member, which also facilitates the coordination of actions.

Furthermore, we presented a study in the BW4T coordination testbed that examined the effects of agents explaining their behavior on coordination in human-agent teams. A first result was that, against our expectations, explanations about agent behavior did not lead to better team performance. In the discussion we suggested several explanations for these results, e.g. the task being too simple. A second result was that, in correspondence to our expectations, humans indicated that they better understood the agent's behavior when they received explanations about it.

Though the BW4T task is simple, we believe that it offers a good platform for investigating human-agent teamwork. In order to further study human-agent teamwork, we intend to do more experiments in the BW4T testbed, in which we will use more diverse conditions than the two described in this paper. We will test the effects of no communication at all and an overload of explanations, and compare them to the current results. We also want to measure more dependent variables in the experiments. Besides time of completing the task, we will also measure the sharedness of knowledge between team members, and the trust humans have in agents. Such research will give insight in whether concepts that were adopted from literature on human teamwork also apply to human-agent teamwork.

In future work, we intend to apply the results of this research to a real world domain. We are aiming for the domain of crisis management, where software agents can support policemen and firefighters by providing information, providing advice, and taking over simple tasks.

Acknowledgments. This research has been supported by the EOARD, grant nr. 103015.

References

1. Austin, J.: Transactive memory in organizational groups: the effects of content, consensus, specialization and accuracy on group performance. Journal of Applied Psychology 88(5), 866–878 (2003)
2. Bordini, R.H., Dastani, M., El Fallah Seghrouchni, A. (eds.): Multi-Agent Programming: Languages, Tools and Applications. Springer (2009)
3. Bradshaw, J.M., Feltovich, P., Johnson, M.: Human-Agent Interaction. In: Handbook of Human-Machine Interaction, pp. 283–302. Ashgate (2011)
4. Bradshaw, J.M., et al.: Representation and reasoning about DAML-based policy and domain services in KAoS. In: Proceedings of AAMAS 2003. ACM Press (2003)
5. Bratman, M.: Intention, Plans and Practical Reason. Harvard University Press, Cambridge (1987)

6. Cannon-Bowers, J.A., Salas, E., Converse, S.: Shared mental models in expert team decision making. Individual and Group Decision Making Current Issues 39(3-4), 221–246 (1993)
7. Cooke, N.J., Salas, E., Cannon-Bowers, J.A., Stout, R.J.: Measuring team knowledge. Human Factors 42(1), 151–173 (2000)
8. Core, M., Traum, T., Lane, H., Swartout, W., Gratch, J., Van Lent, M.: Teaching negotiation skills through practice and reflection with virtual humans. Simulation 82(11), 685–701 (2006)
9. Dennett, D.: The Intentional Stance. MIT Press (1987)
10. Dhaliwal, J., Benbasat, I.: The use and effects of knowledge-based system explanations: theoretical foundations and a framework for empirical evaluation. Information Systems Research 7(6), 243–361 (1996)
11. Gregor, S., Benbasat, I.: Explanation from intelligent systems: theoretical foundations and implications for practice. MIS Quarterly 23(4), 497–530 (1999)
12. Harbers, M., Van den Bosch, K., Meyer, J.-J.: Design and evaluation of explainable agents. In: Proceedings of IAT 2010 (2010)
13. Hindriks, K.: Programming Rational Agents in GOAL. In: Multi-Agent Programming: Languages, Tools and Applications, pp. 119–157. Springer (2009)
14. Janis, I.L.: Victims of Groupthink. Houghton Mifflin, Boston (1972)
15. Johnson, L.: Agents that learn to explain themselves. In: Proc. of the 12th Nat. Conf. on Artificial Intelligence, pp. 1257–1263 (1994)
16. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., van Riemsdijk, B., Sierhuis, M.: The Fundamental Principle of Coactive Design: Interdependence Must Shape Autonomy. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN 2010 International Workshops. LNCS, vol. 6541, pp. 172–191. Springer, Heidelberg (2011)
17. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Hoffman, R.R., Jonker, C., van Riemsdijk, B., Sierhuis, M.: Beyond cooperative robotics: The central role of interdependence in coactive design. IEEE Intelligent Systems 26, 81–88 (2011)
18. Johnson, M., Jonker, C., van Riemsdijk, B., Feltovich, P.J., Bradshaw, J.M.: Joint Activity Testbed: Blocks World for Teams (BW4T). In: Aldewereld, H., Dignum, V., Picard, G. (eds.) ESAW 2009. LNCS, vol. 5881, pp. 254–256. Springer, Heidelberg (2009)
19. Jonker, C.M., van Riemsdijk, M.B., Vermeulen, B.: Shared Mental Models - A Conceptual Analysis. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN 2010 International Workshops. LNCS, vol. 6541, pp. 132–151. Springer, Heidelberg (2011)
20. Keil, F.: Explanation and understanding. Annual Reviews Psychology 57, 227–254 (2006)
21. Lewis, K.: Measuring transactive memory systems in the field: Scale development and validation. Journal of Applied Psychology 88(4), 587–604 (2003)
22. Malle, B.: How people explain behavior: A new theoretical framework. Personality and Social Psychology Review 3(1), 23–48 (1999)
23. Mohammed, S., Klimoski, R., Rentsch, J.R.: The measurement of team mental models: We have no shared schema. Organizational Research Methods 3(2), 123–165 (2000)
24. Moreland, R.L., Myaskovsky, L.: Exploring the performance benefits of group training: Transactive memory or improved communication? Organizational Behavior and Human Decision Processes 82(1), 117–133 (2000)

25. Nandkeolyar, A.K.: How do teams learn? shared mental models and transactive memory systems as determinants of team learning and effectiveness. PhD thesis, University of Iowa (2008)
26. Rao, A., Georgeff, M.: BDI-agents: From theory to practice. In: Proceedings of ICMAS 1995 (1995)
27. Rau, D.: Top management team transactive memory, information gathering, and perceptual accuracy. Journal of Business Research 59(4), 416–424 (2006)
28. Rouse, W., Morris, N.M.: On looking into the black box: Prospects and limits in the search for mental models. Psychological Bulletin 100(3), 349–363 (1984)
29. Sheridan, T.B., Verplank, W.L.: Human and computer control of undersea teleoperators. Technical report, Man-Machine Systems Laboratory, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts (1978)
30. Swartout, W., Moore, J.: Explanation in Second-Generation Expert Systems. In: Second-Generation Expert Systems, pp. 543–585. Springer, New York (1993)
31. Sycara, K., Lewis, M.: Integrating intelligent agents into human teams. In: Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting (2002)
32. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: Proceeding of the International Conference on Data Engineering Workshop. IEEE Computer Society, Washington, DC (2007)
33. van Diggelen, J., Beun, R.-J., Werkhoven, P.J.: Intelligent assistants in crisis management: from PDA to TDA. In: Proceedings of BNAIC 2009 (2009)
34. Van Lent, M., Fisher, W., Mancuso, M.: An explainable artificial intelligence system for small-unit tactical behavior. In: Proc. of IAAA 2004. AAAI Press, Menlo Park (2004)
35. Wegner, D.M.: Transactive memory: A contemporary analysis of the group mind. In: Mullen, B., Goethals, G.R. (eds.) Theories of Group Behavior, pp. 185–208 (1987)
36. Ye, R., Johnson, P.: The impact of explanation facilities on user acceptance of expert systems advice. MIS Quarterly 19(2), 157–172 (1995)
37. Yen, J., Fan, X., Sun, S., Hanratty, T., Dumer, J.: Agents with shared mental models for enhancing team decision-makings. Deision Support Systems 41, 634–653 (2006)

Adaptive Coordination in Distributed and Dynamic Agent Organizations

Kathleen Keogh[1,2,⋆] and Liz Sonenberg[2]

[1] School of Science, Information Technology and Engineering,
The University of Ballarat,
P.O. Box 663 Ballarat, VIC 3353 Australia
[2] Department of Computing and Information Systems,
The University of Melbourne
k.keogh@ballarat.edu.au

Abstract. We elaborate the rationale and design of OJAzzIC (Organizations Joining Adaptively with Improvised Coordination), a model for agents in (Jazzy) Organizations that need to engage in dynamic adaptation to respond to a dynamic situation. OJAzzIC provides an adaptive data structure and framework for creation of multiple instances of organizations within a distributed system, with knowledge sharing across organizational boundaries achieved through overlapping instances.

Keywords: Teamwork, Multiagent Systems, Coordination, Adaptation, Organizations.

1 Introduction

When working in complex dynamic scenarios such as in emergency management or naval navigation, people adjust their own plans to coordinate and fit in with others in order to achieve goals, rather than following strict scripts, protocols or role descriptions [20,27]. Indeed, in complex settings it is not possible to consider all alternatives and create a complete plan, rather an incomplete plan is created based on current knowledge and a sequence of incremental problem solving processes is involved in elaborating this plan, whilst actions begin toward fulfilling the plan. Social scientists have studied ways of designing human organizations to support such improvisation, e.g. [25], in settings that involve uncertainty, incomplete knowledge, changing situations and interdependencies across multiple tasks.

Formal predefined organizations can exist based on structured entities, such as an Emergency Rescue Unit, Military Unit or Service Organization. There are also numerous organizations, sometimes termed *adhocracies* [24] that emerge in a dynamic distributed system [32], due to a local problem (shared location) or a coordination need (e.g. resource contention or shared goal). An adhocracy involves

⋆ Postgraduate student in the Department of Computing and Information Systems, The University of Melbourne.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 38–57, 2012.

multiple groups making decisions in a rapidly changing environment [24,26]. An examination of adaptive organizations found that the complexity and time constraints involved are such that the organization changed via localised adaptive planning and improvisation rather than a broad re-structure at a design level [20].

For similar reasons, it has been recognised that introducing organizational concepts in the design of complex multiagent systems provides capabilities to promote appropriate communication, interaction and coordination, e.g. [18,19,38]. In dynamic domains, the interaction and coordination cannot all be pre-scripted, but must be adaptable at runtime, so that as situations change, the collective of agents can change goals and reallocate tasks or collaborate on tasks in response to availability changes. We are working towards the design of a flexible, coordinated organization-based agent system comprising multiple agents working toward a shared goal. We are not only looking at plan elaboration, the OJAzzIC (Organizations Joining Adaptively with Improvised Coordination) model also supports appropriate knowledge transfer within and across organizations and obligations to ensure relevant knowledge is shared, and is intended to provide a framework to enable coordinated, improvised activity.

The capabilities we expect of our sophisticated agents in order to cope in a complex, uncertain and dynamic environment are based on problem settings with the following characteristics:

- Multiple agents are working with at least one high level shared objective;
- Agents work with individual rationality (self-interest and individual utility function) *as well as* some form of group rationality;
- Interaction, coordination and cooperation between individuals or groups is needed in order to achieve goals with interdependencies;
- Membership of groups is fluid — agents may come and go;
- Roles are not fixed — members are required to improvise in order to achieve goals in a timely way — based on who is available and their capabilities;
- The problem is distributed across multiple locations, central coordination and control is not possible.

These capabilities require a sophistication in terms of agent knowledge, group knowledge and awareness. The OJAzzIC model is based on an organizational approach. Using the structure of an organization, agents have contracts that define how knowledge is shared and held consistent between agents.

We consider an individual agent to have individual mental attitudes such as beliefs, goals, intentions and plans to enact those intentions, and accordingly we build on the traditions of the Beliefs, Desires and Intentions (BDI) architecture for individual agents, extended to group activity, e.g. [38]. We adopt an agent organization as defining a structure for a group of agents with some shared mental attitudes associated with the organization, in addition to individual attitudes. The agent organization is a structured group of agents with definitions of roles defining responsibilitie and relationships between roles, and rules defining

obligations on members. For some, an agent organization is a group whose roles and interactions are typically expected to be relatively stable and change slowly over time [29]. Others have introduced terminology of groups, teams, congregations, coalitions etc, and these are variously associated with properties of coordination, correlation, cooperation, and other Co-X words [18,31], and levels of autonomy between individuals and the group [4,9,15,34]. In this paper we adopt a rather relaxed view, and simply use the term organizations to cover both longer term, stable groupings and those that can be relatively short-lived and changed via localised adaptive planning.

In this paper we target a structure enabling both reallocation of agents to tasks and dynamic goal decomposition and achievement. Previously, we examined SharedPlans [16] in the context of human coordination in the emergency management domain and highlighted that agents need to ensure that they cultivate knowledge about their organizational structure as well as domain knowledge — plans and situation awareness [21]. In future work, we hope to give attention more directly to the management of interdependent resources. These requirements are not unique to the emergency management domain and could be relevant to many emergent situations where agents initially form or enlist in an organization with a common goal, then within that organization, smaller organizational groups form to autonomously work on distributed, but possibly interdependent sub goals. The issue to highlight is that each organization needs to be aware and coordinated within the organization and across any overlapping organization.

We are particularly interested in awareness and coordination of knowledge and behaviour between, across and within organizations. In this paper, we present a high level design for our organizational approach and describe how multiple dynamic instances of organizations are created in order to enable appropriate awareness in the organization. We do not yet address important issues including: policy creation, agent negotiation protocols, resource contention and processes for creation of and disbanding of organizations.

The paper is structured as follows. In the next section, we present key background material: we describe an extension to the traditional agent BDI deliberation cycle to include two levels of agent awareness of others within an organization — i.e. agents not only consider other agents in their own individual deliberations, but agents also deliberate with others in an organization; in subsection 2.1, we describe a scenario involving detection robots to highlight key requirements for our design; and in subsection 2.2 we summarise the main elements of the model OMACS [6,7,8] on which we build. OJAzzIC builds on the adaptable organizational structure used in OMACS [7] and combines with features from contract based systems [10] and intentional approaches to joint planning [16]. Section 3 contains a discussion of the key ideas of our model, including features, metamodel, and a discussion of goals and roles. In section 4, we briefly cover related work on adaptive agent organizations. We conclude by highlighting future directions of our work.

2 Background Material

2.1 Motivating Scenarios

A robotic search for weapons of mass destruction scenario previously used in adaptive agent system design will be outlined and then we will modify this scenario to highlight some of the requirements we are addressing. This scenario was used previously to describe a simple adaptable organization based on OMACS (Organization Model for Adaptive Computational Systems) [7].

The scenario is based on a number of robot sensors that need to search an area and identify suspicious objects. Each suspicious object needs to be checked to see if it is a biological, chemical or radioactive weapon. Not all sensor robots have the capabilities to perform each object identification. When a suspect object is found, all 3 checks need to be performed until one matches. So, each weapon may only be one type of weapon, and the checks may be performed in any order. There are six agent types: Base Robot, Sophisticated Robot, Chemical Robot, Biological Robot, Nuclear Robot and Remover Robot. Both the Chemical Robot and the Sophisticated Robot can identify chemical weapons, but the Sophisticated Robot's chemical detector is not as good as the Chemical Robot's chemical detector. All robots can search and find suspect objects. When a weapon has been successfully identified, the Remover Robot removes it.

We now propose three modified scenarios to highlight more complex requirements for coordination:

Scenario 1. Goals involving multiple agents:

Example 1a. Suppose the chemical weapon can only be detected successfully by two robots working together simultaneously — Chemical Robot and Sophisticated Robot. The two robots need to coordinate their behaviour to both move to the same object simultaneously in order to perform the detection task.

Example 1b. Suppose two agent types Base Robot and Chemical Robot, can combine their individual capabilities in order to achieve the removal role capabilities of one Removal Robot agent. In this case, if an agent Removal Robot fails or leaves the scene and another Removal Robot is not available, these two agents could together combine to achieve the tasks that were previously allocated to one agent: Removal Robot.

Scenario 2. Resource contention and interdependencies: Suppose the removal robot agents require an additional resource: a trolley, to help remove the detected weapons. Each removal robot has access to at least one trolley, shared by other removal robots within a close proximity. Sharing of this resource requires that the agents coordinate their use and movement of the trolley. As agents move about, they may need to become aware of 'new' trolleys closer to them.

These modifications highlight issues that we are interested in addressing in the context of agent organizations. We focus on situations where tasks require multiple agents acting in a coordinated way to complete them, and therefore we

require organizations of agents who share a goal. Within the organization, agents can work autonomously on some tasks, but must coordinate where necessary. In some cases, there may be a commander or leader role, however sometimes the coordination may be established directly between members of the organization. Agents should be able to dynamically reorganize and reallocate tasks if agents leave or become unable to fulfill responsibilities.

Based on an analysis of human coordination in Emergency Management [21], we seek to address the following requirements for coordination in a dynamic organization:

- *Awareness.* All players in the organization are obliged to work with awareness of others, including to be aware of what is relevant to others.
- *Appropriate knowledge sharing.* Information, relating to a goal or resource, may flow within and out of one organization into other organization(s) as members identify relevance to other organizations that they belong to, or are aware of.
- *Flexible adjustment of behaviour.* In a dynamic situation involving uncertainty, agents adjust their behaviour to fit in with others. The action sequence emerges over time based on the situation and adaptation to address changes as they are realised. Goals may also change.

These requirements apply to complex domains that demand flexibility due to uncertainty, distributed knowledge and interdependencies that must be managed. Agents need to be able to improvise — adjust plans and modify goals to fit in with others. It is not possible to prescribe behaviour exactly for all circumstances at design time so agents need to have access to organizational knowledge to enable reasoning to change individual and organizational goals and plans.

2.2 OMACS

OMACS (Organization Model for Adaptive Computational Systems) has been developed as an organization design based framework model that is capable of adaptation so that a system organization can organize and reorganize itself dynamically at run time. The organization is defined to include the following entities: Goals, Roles, Agents, Domain Model and Policies [6,7,8]. OMACS addresses some of our requirements for flexible and adaptive organizations and has influenced our design.

In OMACS, Deloach introduces the *Capabilities* abstraction to enable flexible and dynamic reallocation of agents to roles [7]. A Role Model is used to define a list of tasks or responsibilities to be fulfilled. Each Role definition within the Role Model includes a required Capabilities list as well as a function that enables capabilities to be prioritised as to their relative importance. This function enables the measurement of an agent's utility to play each role. It is possible to define multiple alternative roles that are capable of achieving a particular goal, however only one role can be allocated to one goal at any one time. DeLoach and colleagues have proposed that adaptability in planning can be addressed by having alternative paths available in a goal decomposition.

When agents are no longer available and goals cannot be met according to the original goal-role-agent assignment, the system automatically reorganizes and newly revised roles or goals are selected based on the currently available agents' capabilities [7]. However, if there is no agent available with an exact match to the capabilities for a required role, the OMACS system does not address this situation. It does not provide for dynamic flexibility at the level of coordination of multiple agents together performing one role to achieve a goal.

In OMACS, most of the information required for collaboration between roles is embedded in the goals that are instantiated [7]. OMACS is associated with the creation of a dynamic goal design and run time representation using Goal Model for Dynamic Systems (GMoDS) [6]. Using careful goal design according to a goal decomposition tree, goals can be ordered so that goal dependency is represented in the goal model, as well as alternative options, so that if one set of goals cannot be satisfied, an alternative set may be chosen. This enables dynamic and flexible re-allocation of goals. The capabilities abstraction enables flexibility in dynamic allocation and re-allocation of agents to roles based on a changing context. However, coordination between agents is implicit in the plans that individual roles have to enact in order to achieve autonomous goals. OMACS does not provide explicit coordination mechanisms toward agent cooperation on goals or cooperation on synchronizing loosely coupled activities so as not to interfere with other agents. If two agents need to act together concurrently, they would need to each have separate goals for the actions and coordination would be achieved implicitly according to the predefined script or plan for each goal that each agent follows autonomously.

2.3 Extending the BDI Agent Deliberation Cycle

Our approach is based on the popular BDI architecture that enables goal-directed behaviour in the presence of explicit deliberation about changes in the environment, but extended to accommodate reasoning about other agents e.g. [38]. Traditional BDI agents deliberate based on a self-interested cycle, i.e.

Table 1. Individual Agent BDI Deliberation Cycle

```
repeat
        perceived-events := event-selector(event-queue);
        update-attitudes();
        plan-options := option-generator(perceived-events,current-goals);
        selected-plan-options := deliberate(plan-options);
        update-intentions(selected-plan-options);
        execute();
end repeat
```

As Corkill [5] and others argue, agents within an organization, need also to consider organizational objectives in addition to their own goals. Agents must hence ensure that individual mental attitudes are managed so that they are not inconsistent with organizational attitudes. Corkill describes organizationally

adept agents that use adjustable preferences to value self, others and the organization to different degrees whilst evaluating utility functions in selecting actions to perform. We propose that this be further extended, so that agents within an organization might not only deliberate individually, but as an organization. Agents therefore would be capable of individual utility based (self-interested) reasoning, other-centred reasoning — with *awareness of* others, as well as organizational reasoning *with* others. In our approach, the obligations to update individual mental attitudes are made explicit in a social contract that defines congruence between organizational attitudes and individual attitudes.

Table 2. Organizational Agent BDI Deliberation Cycle

```
repeat
        perceived-events := event-selector(event-queue);
        process-individual-and-organizational-attitudes(current-social-contracts):
        plan-options := option-generator(perceived-events, current-goals);
        selected-plan-options :=
                        deliberate(plan-options, other-agents-and-organizations);
        update-intentions(selected-plan-options);
        execute();
end repeat
```

In the extended deliberation cycle, an agent perceives environmental input, updates individual mental attitudes but, before selecting an intention the organizational agent will deliberate *with others in the organization.* This organizational deliberation is defined by obligations and policies in the social contract within the organization and is discussed further in Section 3.4. Following the first stage of organizational deliberation and any needed adjustment to individual attitudes, the agent continues the deliberation process *considering others* — when the agent may revise individual intentions and plans to ensure that they are not intending anything that will hamper others and possibly to add new intentions to help others.

3 OJAzzIC: Agents in Organizations Joining Adaptively with Improvised Coordination

In this section, we outline our model, OJAzzIC and how this is used to instantiate a network of multiple coordinated organizations. This model is for Organizations Joining Adaptively with Improvised Coordination (OJAzzIC), the adaptive requirements result in a model that can capture the necessary static and dynamic knowledge in such as way as members can behave as a Jazz musician might — to improvise and adapt their script on the fly, but not in such a way that it would interfere with the script or plan adopted by others. The plans need to be clear, but flexible. Behaviour needs to be coordinated, but not prescribed. OJAzzIC builds on the adaptable organizational structure used in OMACS [7] and combines with features from contract based systems [10] and intentional approaches to joint planning [16].

3.1 Adaptability in Design — Features of OJAzzIC

The novelty in our approach is to combine the adaptive nature of a dynamic organizational structure—enabling both reallocation of agents to tasks and dynamic goal decomposition— with a dynamic social contract that defines explicit obligations and coordination policies on the fly. The social contract may be based on a predefined script that can be well defined or loosely governed by predefined landmarks [40]. Additionally, our proposal for use of multiple *instances* of overlapping organizations in a dynamic way enables adaptable coordination within meta organizations.

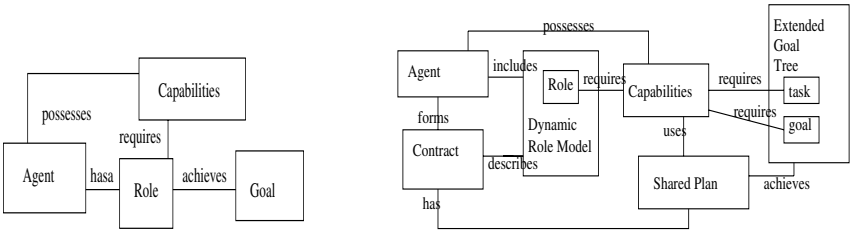
An organizational instance is created dynamically whenever a complex problem arises that requires some coordination over time. This coordination may involve an emergent plan that needs revision when further information becomes available. It may also require coordination of members in terms of a shared resource or goals that require multiple members to coordinate activity dynamically in order to achieve the goal. It may also be that some initiative will be required in terms of using members outside of their usual role descriptions where they have capabilities toward helping achieve a goal.

Key to the approach of OJAzzIC is the creation, as required, of dynamic instances of organizations. Within each organizational instance, context specific dynamic contracts define agent allocations, obligations and roles. These ensure that coordination, knowledge sharing and behavioural obligations can be dynamically defined within a particular network (organization) of agents. Multiple organization instances may be created within the original organization. Each organization has at least one goal that requires coordination. Each organizational instance (including the high level organization) has a set of dynamic organization attributes with values available to all members. These attributes include an organizational structure (role model including dynamic role definitions and coordination roles), set of agents, goal tree, domain beliefs, resource list, fixed domain policies and role definitions and a dynamic contract. The contract defines the allocation of agents to tasks/coordination roles and dynamic coordination and knowledge-sharing policies to ensure consistent beliefs are maintained within the organization.

In order to address our requirements and establish an organizational design with the flexibility to adapt, the following major decisions were made: The model would include agentified organizations as first class entities [38]; Agent-Role-Task mapping using Capabilities would be used to enable flexible automated reallocation [7,36]; Goals could be shared by multiple roles using Capabilities and Tasks; and Organization instances created would include social contracts to define coordination obligations dynamically [40].

Agentifying the organization means we can treat the organization as one agent, with mental attitudes that can then be semantically related to individuals in the organization as desired [38]. This also means that no one individual needs to stay in a particular role (e.g. Leader) for appropriate communication with the organization. The organization is addressable as an agent in its own right [28].

The organization is a static predefined structure that may be instantiated by an actual organization instance that is created at run time. This organization instance may be quite a stable and permanent structure based on formal roles, but it may also be a short term organization created so that a group of agents can work together in a coordinated way. In the latter case, the coordination may be negotiated dynamically rather than be based on predefined scripts. In the former case, the coordination may be based on default scripts, though these can still be adapted. All organization instances are considered to be dynamic, first class, agentified entities and from here on, we shall refer to these as organizations. The organizational contract is part of the organization's knowledge, so accessible to all agent members.



(a) Agent-Goal Abstraction in OMACS

(b) Agent-Goal Abstraction in OJAzzIC

Fig. 1. Comparison of Agent to Goal Relationships in OMACS and OJAzzIC

Figure 1 shows an abstract high level view of the relationships in an OJAzzIC organization alongside a partial view of similar components in an OMACS organization [7]. Figure 1(b) is expanded upon in Figure 2. In Figure 1(a) Agents are related to Goals using the Role abstraction. As discussed in section 3.3, this assumes that one role will achieve one goal. We introduce an extra level of separation between roles and goals. In OJAzzIC, the Goal Tree is extended to include Tasks as a possible decomposition of Goals. Agents can be allocated based on responsibilities for Roles or based on Capabilities to fulfil Tasks. Extending Goal Trees using Tasks will be discussed further in section 3.3.

The Role Model in OMACS is a fixed relationship of predefined roles (represented as 'Role' in Figure 1(a)) whilst in OJAzzIC (Figure 1(b)), the Role Model is dynamic, it is created based on context and represents the roles instantiated by agents in the organization. The Contract is an explicit mental attitude adopted by the organization and defines obligations regarding knowledge sharing and will be discussed further in section 3.4. Figure 1(b) is a simplified conceptual model, more detail is presented in Figure 2. Our design is motivated by reality. In real situations, a role description may change based on context and roles might need to be shared. In OJAzzIC, the dynamic Role Model enables goals to be shared between multiple roles and where necessary coordination roles are created. The structure of the organization — role relationships and role definitions — are dynamically defined.

We are not alone in proposing the need for shared mental models. Commitments toward maintaining and proactively sharing information in teamwork has

been addressed in similar work with agent/human teams [41]. In order to establish information relevancy, explicit information-needs graphs have been used along with explicit mental models of team structure, team processes, and domain knowledge [12]. Information flow within groups has been described in terms of the relationships that form in a coordination loop [32]. In our case, we propose that encouraging information sharing in each such network requiring coordination is possible if each is considered a dynamic instance of an organization. Within each organizational instance, obligations exist ensuring appropriate knowledge sharing. Each organizational instance has a shared goal — such as a knowlege seeking goal or a goal to manage a dependency, or a goal to achieve a particular set of tasks. Agents may belong to multiple organizational instances simultaneously. In this way, an agent's knowledge can be shared across organizational boundaries where it is relevant to more than one organizational group.

3.2 OJAzzIC Organization Model

Figure 2 shows the OJAzzIC organizational model. Each organizational instance is created following this model. Our organization entity is loosely based on OMACS [7], with extensions to provide for more flexible and dynamic goal/role sharing. Where we adopt OMACS concepts without extension, we do not provide details here, but direct the reader to details elsewhere [8].

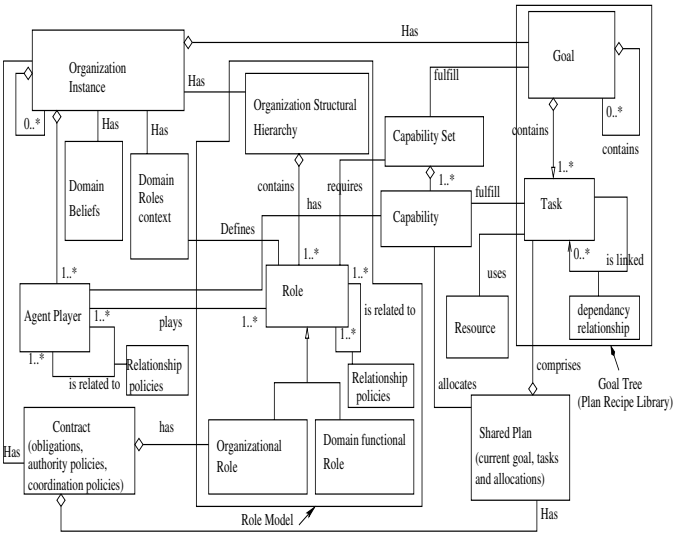


Fig. 2. The OJAzzIC organization model

To achieve goals the organization may require more than one agent to coordinate behaviour for related tasks. Goals are described in terms of tasks and tasks require capabilities. A player has capabilities, enacts a role and is also allocated tasks based on that role. Plans are based on instantiating general default

plans or creating dynamic plans based on the decomposed Goal Tree, according to the particular context. Organizational plans to achieve goals are established dynamically using SharedPlans [16].

Agents start by belonging to a large organization responsible for the main high level goal (e.g. the entire system). When two or more players need to coordinate, a new organization is formed and within that organization an explicit contract is formed that dictates policies, obligations, agreed goals and agreement to coordinate with others in that organization. Coordination within the organization relies on appropriate communication to share relevant information and share plans. The new organizations that form overlap with existing organizations. We propose that in a dynamic organization, multiple smaller organizations are created. These organizations each need to be explicit so that appropriate coordination can be established within each. Each organizational instance created would be based on the OJAzzIC organizational entity structure.

In OJAzzIC, an organization O is a tuple:

$<G*, R, Re, Contract, A, C, P, \sum, \beta,$ *oaf, achieves, requires, possesses* $>$

This extends OMACS with G*, R, Re and Contract. These are explained in more detail in the subsequent sections.

G*: extended Goal Tree, including ordered tasks where possible. This defines the goals and how they could be decomposed into sub goals and ordered tasks;

R: the Role Model including a set of Roles, relationships between Roles and context based Role Definitions. This also includes coordination roles created dynamically as necessary;

Re: a dynamic Resources list defining objects in the environment that can be used to help perform tasks;

Contract: The dynamic Contract contains a social contract and an information contract. The social contract comprises a SharedPlan [16] and a set of coordination Roles agreed for the organization. The Shared Plan outlines the current selection of tasks to achieve the goal and the allocations thus far assigning responsibilities for tasks. The contract in OJAzzIC replaces ϕ in OMACS (ϕ is a relation over G x R x A providing goal/role/agent assignments). The information contract is a set of agreed policy obligations and commitments to intentions to ensure consistency of beliefs within the agent organization. The information contract contains β the current Beliefs set that includes beliefs about the environment, including resources.

A: set of Agents;

C: set of Capabilities;

P: fixed policy constraints to apply to all members and to the allocation of tasks;

$\sum$: domain model used to specify environment objects and relationships

In OMACS, policies are abstractly used to define the processes for allocation of agents to roles (Assignment Policies), define behavioural obligations and relations between roles (Behavioural Policies) and define structural reorganizational processes such as how to reallocate tasks (Reorganization Policies). OMACS defines policies that must be held as *Law* policies and policies that can be prioritised and hold when possible as *Guidance* policies. OMACS also defines additional supporting functions *oaf, achieves, requires, possesses.* The definition of function achieves has been extended to include tasks and SharedPlans

oaf: organization assignment function measures the utility of a particular SharedPlan assignment of Agents to Roles to Tasks;
achieves: function defining role assignment — how effective the behaviour of roles can be to achieve task T or goal G in a SharedPlan. This is used if a default plan needs revision or if a default plan cannot be found for a context. The Goal-Tree can be used to derive a plan.
requires: defines the capabilities required to play a role R or task T; and
possesses: function defining the quality of an agent's capability for a particular Task. To decide how well an agent can play a role, the requires and possesses functions are combined into a function: capable. To decide how well an agent can play a role to achieve a goal, the capable function and achieves function are combined as a function: potential.

Based on our requirements, in OJAzzIC, we incorporate additional *Authority* Policies defining a process for explicit acceptance of allocations by an agent as well as *Coordination* Policies to help resolve multi-agent plan coordination dynamically.

3.3 Goal Trees and Dynamic Role Model

In order to achieve our aim of flexibility, we choose to keep separate the goals of an organization and the available roles that may be used to define (or allocated) responsibility to achieve these goals.

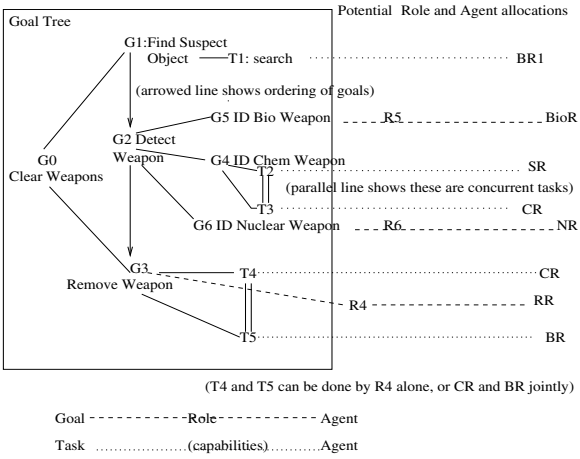


Fig. 3. OJAzzIC Goal Tree showing potential allocations with Players

Synchronized Tasks. If a goal cannot be achieved by one role, then multiple roles or agents can combine to achieve a goal by working together. We describe goals as composing synchronised tasks and tasks can be performed by agents with the appropriate capabilities. This abstraction is introduced to enable

flexible and dynamic planning by agents to establish a coordinated SharedPlan to work together to achieve a goal. We choose to split goals into separate synchronised tasks rather than split roles, as intuitively, this abstraction fits with our observations from real life examples in the Emergency Management domain.

In figure 3 an example OJAzzIC Goal Tree for the sensor case from section 2.1 is shown, as defined at design time. The Goal Tree encapsulates knowledge about goal decompositions and in this case, we have indicated on the right side how the goal tree could be expanded at run time to link to capabilities of particular agents and generate potential plans. The Goal Tree can be thought of as a plan recipe library. Default or preferred plans can be defined by indicating preferred paths at design time, however the flexibility to use the Goal Tree dynamically allows for dynamic planning and revision of plans.

As figure 3 shows, a goal may comprise multiple sub goals. Goals may also be decomposed into tasks. Tasks and goals can be ordered. This abstraction is to enable the splitting of goals to share between multiple players. When a goal is split, performing the tasks requires coordination between the players. In figure 3, goal G0 is decomposed into sub goals G1, G2 and G3. G1 must be performed before G2. G1 is described in terms of search task T1. In some cases, one goal can be achieved by one role directly, as with G3 achieved directly by R4. In the absence of an agent allocated to Role R4 for whatever reason, an alternative is to split goal G3 into two separate tasks T4 and T5. These tasks then can be allocated to individual agents based on individual capabilities. When goals are split and shared then the agents need to coordinate their behaviour using a SharedPlan. In figure 3, different dashed lines indicate whether an agent is allocated to perform a role that directly satisfies a goal (big dashes) or whether agents are allocated individual tasks (small dotted line). In the latter case, the SharedPlan will ensure coordination between the agents. This figure shows potential allocations of players (sensor robots) to these goals and tasks at run time. BR1 has the capabilities to perform task T1 and in this example has been allocated that task. Player RR has the capabilities to fulfil role R4 and thus could be allocated in that role to perform both tasks T4 and T5. Alternatively, CR could perform T4 and BR could perform T5. Tasks T4 and T5 must be done simultaneously — indicated by the parallel lines connecting them.

Using the sensor agent case as a very simple example, we can identify that if allocations were made based on Figure 3 with CR and SR working together to achieve the goal G4 Detect Chemical Weapon, then CR and SR would need to coordinate and create a SharedPlan. This is a simple instance of an organization. Creating an organization would then obligate each agent (as defined in the social contract) to appropriately share information such as a SharedPlan to facilitate coordination between these agents. There could be initially an organization involving BR1 and perhaps some other agents conducting a search of the area. Then as suspect objects are found, new organizations could spring up including agents able to detect weapons. For example, an organization with BioR, SR, CR, NR could form with BR1 as a leader to achieve the goal, G2 Detect Weapon. Then when a weapon has been identified, a new organization

involving these or other agents would form to remove the weapon. We could imagine a more complex scenario where there were hundreds of agents involved in the weapons search over a large area. Multiple organizations could be created dynamically as agents elect to work together to achieve goals. When one agent is part of multiple organizations, this overlap allows for relevant information (e.g. location, detection status of object) to be propagated across the network across organizational boundaries.

Resources are explicit entities in the OJAzzIC organization's knowledge base. This is because of the potential for resource contention amongst agents in the organization and thus the need to coordinate interdependencies. Having this knowledge explicit will enable direct reasoning within the Agent Organization, based on priorities, negotiated protocols or the use of coordination artifacts [30,1]. We do not address such reasoning further in this paper.

The sharing of relevant knowledge within and across organizations is important as knowledge may be distributed. Obligations to ensure appropriate knowledge-sharing about SharedPlans [16] has been well defined and we adopt this intention based approach to planning. We have adopted the use of contracts to manage obligations to ensure agents will share domain-knowledge as well as structural and coordination knowledge within each organization. Collective obligations can be implemented as policies to govern joint activity and teamwork [39]. We leave implementation details aside and in the next section, describe at a conceptual level the contents of the contracts that need to be created.

3.4 Contracts

As agents join (or apply to join) an organization, then agents must agree (commit) to a social contract that defines interaction within the organization. Beliefs, values, objectives, protocols and policies may be defined in the context of the social relationships that exist within the society. At the time a new organization is instantiated in OJAzzIC, players in each organization, explicitly form an organizational contract. Each organization exists for the duration of time in which there is a need for that group of agents to be coordinated. The organizational contract comprises a social contract that defines the social structure of the organization and an information contract that defines how information is shared within the organization.

The *social contract* defines role descriptions and agreed role allocations. Role descriptions may be abstractly defined at organizational design time and adapted dynamically. Roles are defined with associated capabilities, authority levels and obligations. These are made explicit in the social contract to enable dynamic and adaptive revisions. Having an explicit social contract provides the abilitiy to predict others' behaviour and flexibly adapt individual goals in anticipation of others' needs and behaviour. The social contract also defines an agreed model for command, control and coordination. Coordination Roles such as Leader, Resource Manager, Knowledge Manager and Contract Manager are identified and allocated if needed.

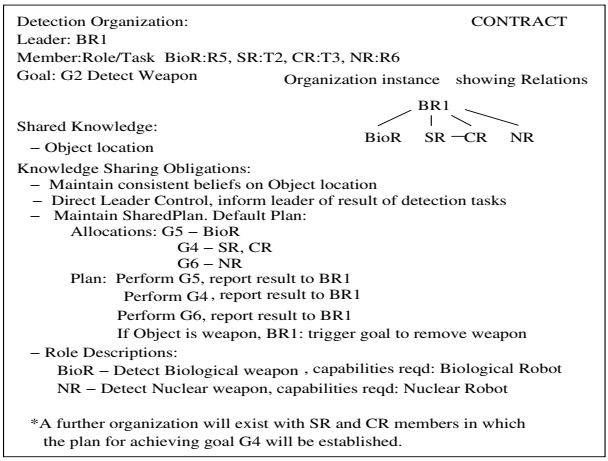


Fig. 4. Partial contract for the detection team in the sensor robot case scenario

Based on the need to cultivate knowledge sharing [21], *information contracts* include policies that obligate members in an organization to all adopt joint intentions to cultivate mutual knowledge within the organization. Obligations to share information are limited to the agents within each individual organization that forms. As an agent can belong to multiple organizations, the overlap enables relevant information to be dispersed across a wider network as necessary. Figure 4 shows parts of an example contract for the organization with goal G2: Detect Weapon, in the sensor robot case study.

4 Related Work

We have looked particularly at the following adaptive organizational models for agents: OMACS [7], KB-ORG [36], and OperA [11]. Each addresses part of our requirements. We also consider adaptivity in relationships achieved by associating Agents to Roles and Goals as discussed, for example, in previous work by Ferber [13] and Odell [28].

In these proposed metamodels for Agents, Roles and Groups [13,28], within a group context, agents are associated with an agent role to determine the sorts of activities in which the agent may participate. The interactions between agents are governed by the roles played by the agents. An agentified group can then communicate, take on a role and act as an agent. Roles enable a layer of abstraction between agents and their allocated tasks facilitating re-allocation and reorganization of agent groups. Tidhar has similarly defined an organization as a set of related teams as a first class (agentified) entity [38].

We have looked for flexibility in our goal design and have extended the OMACS definition, so that a goal might be broken into synchronized/ordered

tasks that could be assigned to agents. This approach has also been adopted in AGR [13] and in the MOISE system [14] where a distinction is made between a separate structural specification and a functional specification. In MOISE-Inst [19], goals are decomposed into missions, then allocated to a set of responsible agents. The goal tree specifies potential tasks that can be associated with individual plan recipes that achieve each leaf goal [18,36].

A related approach has been promoted in the KB-ORG system, designed for automatic allocation of tasks to agents in a dynamic organization [36]. In KB-ORG, roles contain an assignable list of responsibilities and if necessary, roles can be split between a set of agents and explicit coordination roles are created. KB-ORG differs from our approach in that we are attempting to create a design for organizationally aware agents, rather than using an external management system. Importantly, in order to establish dynamic coordination by agents in organizations, explicit coordination roles are needed. In KB-ORG, coordination roles are created when an application level role is split between a set of agents. Similar coordination roles are adopted in the role model and agreed in a social contract in OJAzzIC.

We are not unique in articulating agent interactions as requirements and modelling these separately in the design process for an agent system [33]. Others have described interactions as part of an organizational design [2,14]. Relationships and awareness of relationships between agents in a dynamic organization are important to enable the appropriate coordination and communication.

Functional specification and decomposition of tasks in MAS using a goal-tree to specify tasks with synchronization or coordination relations is not unique. It is found in models including for example STEAM [37] and TAEMS [23]. High level guidelines have been used to describe constraints on how organizational objectives should be decomposed in a hierarchy. Separately, operational objectives represented as leaf goals in their goal decomposition can be operationally coordinated as required by the individuals involved (not at an organizational level) [5,36]. This abstraction to 'leave the details' to the smaller groups is similar to ours, although we make the distinction that these smaller groups may be considered as temporary organizations rather than teams. The value in creating short to medium term organizations, is that for the duration of the organization, obligations and some infrastructure including shared mental attitudes, can be used to help ensure that our complex, dynamic, coordination requirements may be addressed. A separate approach is to use Petri Net models to monitor and coordinate hierarchical team plans [3].

Coordination by proxies or intermediate layers within an agent architecture has been suggested to enable open systems with heterogenous agents to work together. For example, Scerri *et al* assign a proxy agent responsible for coordination to each team player [35]. This enables domain specialised agents to work as part of a larger team, without the need for knowledge about the team itself. However, with this approach, the agent players are not able to directly reason about team issues or coordination and this limits its applicability in our context.

5 Conclusion

We have provided an elaborated description of a model for agent organizations requiring adaptability and improvisation. By giving agents access to organizational information that they can change, we allow agents to adjust their own attitudes to fit in others in a changing situation. We have taken features from existing systems [7,11,16,36] and extended these to meet our requirements.

We have organizations as first class entities so membership does not need to be fixed and members have access to the mental state of the organization. Organizations are created based on a need to coordinate actions or resources. Within organizations, agents are obliged to share information and maintain consistent plans. We have contract based dynamic organizations so that organizational structure is defined/agreed and modifiable at run time. We have a flexible goal-task decomposition that enables definition of concurrent tasks and goals that require multiple agents. We allow for goals to be shared, where multiple agents can combine their capabilities to collaborate to achieve a goal. We propose that this model be implemented so that multiple organizations are created, as needed. As the OJAzzIC system is an organization of organizations, reasoning at different levels of abstraction is possible.Each organization manages obligations to ensure relevant knowledge is shared within and between organizations.

Future work is needed to formalise this design and validate it with an implementation. We hope to test it with a multi-agent organization implementation based on the modifications to the search for weapons discussed early in this paper [7]. Ultimately, our intent is to use our model in organizations involving agents and humans, c.f. [41].

Agents may leave or join an organization at any time. Upon creation or joining of an organization, agents agree to obligations within the organizational contract. In particular these obligations guide communication and shared intentions to keep mutual knowledge consistent between members. Agents may have individual utility functions for private goals as well as global utility functions at an organizational level. Agents may also use initiative when they have capabilities outside of their designated role to locally adapt the organizational structure to assist by performing tasks in the interest of the organization. These policies and obligations need to be formally specified in future work.

OJAzzIC organizations would be suited to work in highly dynamic, complex domains that require flexible adaptive interactive behaviour. These could include Emergency Management systems, Naval management coordination and to some extent military command and control (when local decision making occurs separate from the formal vertical command hierarchy). Where short term coordinated tasks are to be performed by a group of agents and are well specified, not likely to change during execution of a plan, an organizational structure is not necessary. In these cases, agents might form a different less structured collective such as a group with a SharedPlan [17].

Multiagent systems enabled with characteristics to work with humans have potential benefits in simulation and training. Sophisticated agents could potentially be used as surrogate humans in virtual organizations for training exercises.

To be useful, such agents need to behave in a predictable and believable way similar to humans [22] and need to be designed to coordinate behaviour with other players. Our work contributes by proposing a conceptual design for dynamic and adaptable agent organizations working together in a coordinated way.

Acknowledgements. The authors thank Dr Gil Tidhar for his conversations and suggestions regarding OJAzzIC. Also, thanks to the participants in the COIN workshop at AAMAS 2011 for their feedback and to the anonymous reviewers for their constructive suggestions.

References

1. Acay, D., Tidhar, G., Sonenberg, L.: Extending agent capabilities: Tools vs. agents. In: Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 259–265 (2008)
2. Argente, E., Julian, V., Botti, V.: MAS Modeling Based on Organizations. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 16–30. Springer, Heidelberg (2009)
3. Bonnet-Torrès, O., Tessier, C.: A Formal Petri Net Based Model for Team Monitoring. In: Dignum, V. (ed.) Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models, pp. 568–589. IGI Global, Information Science Reference (2009)
4. Brooks, C., Durfee, E.: Congregation formation in multiagent systems. Journal of Autonomous Agents and Multiagent Systems 7, 145–170 (2002)
5. Corkill, D., Durfee, E., Lesser, V., Zafar, H., Zhang, C.: Organizationally Adept Agents. In: COIN Pre-Proceedings 12th International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN@AAMAS 2011), Held in Conjunction with 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), pp. 15–30 (May 2011)
6. DeLoach, S.A., Miller, M.: A goal model for adaptive complex systems. International Journal of Computational Intelligence: Theory and Practise 5(2) (2010)
7. DeLoach, S.A.: OMACS: a framework for adaptive, complex systems. In: Dignum, V. (ed.) Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global, Hershey (2009) ISBN: 1-60566-256-9
8. DeLoach, S.A., Carlos, G.-O.J.: O-MaSE: A customizable approach to designing and building complex, adaptive multiagent systems. International Journal Agent Oriented Software Engineering 4(3) (2010)
9. Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. PhD Thesis, Universiteit Utrecht (2004)
10. Dignum, V., Meyer, J., Weigand, H.: Towards an organizational model for agent societies using contracts. In: Proceedings of AAMAS 2002, First International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna, Italy (2002)
11. Dignum, V., Vázquez-Salceda, J., Dignum, F.: OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 181–198. Springer, Heidelberg (2005)

12. Fan, X., Sun, S., Sun, B., Airy, G., McNeese, M., Yen, J.: Collaborative RPD-enabled agents assisting the three-block challenge in command and control in complex and urban terrain. In: Proceedings of 2005 BRIMS Conference Behavior Representation in Modeling and Simulation, pp. 113–123. Universal City, CA (2005)
13. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
14. Gâteau, B., Khadraoui, D., Dubois, E., Boissier, O.: MOISEInst: An organizational model for specifying rights and duties of autonomous agents. In: Proceedings of Third European Workshop on Multi-Agent Systems (EUMAS 2005), pp. 484–485. Elsevier Science B.V. (2005)
15. Griffiths, N.: Supporting cooperation through clans, cybernetic intelligence - challenges and advances. In: Proceedings IEEE Systems, Man and Cybernetics, 2nd UK&RI Chapter Conference (2003)
16. Grosz, B., Hunsberger, L.: The dynamics of intention in collaborative activity. Cognitive Systems Research 7(2-3), 259–272 (2006)
17. Grosz, B., Kraus, S.: The evolution of shared plans. In: Foundations and Theories of Rational Agency, pp. 227–262 (1999)
18. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(4), 281–316 (2005)
19. Hübner, J., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents 'giving the organisational power back to the agents'. Autonomous Agents and Multi-Agent Systems 20, 369–400 (2010)
20. Hutchins, E.: Organizing work by adaption. Organization Science 2(1), 14–39 (1991)
21. Keogh, K., Sonenberg, L., Smith, W.: Coordination in Adaptive Organisations: Extending Shared Plans with Knowledge Cultivation. In: Vouros, G., Artikis, A., Stathis, K., Pitt, J. (eds.) OAMAS 2008. LNCS (LNAI), vol. 5368, pp. 90–107. Springer, Heidelberg (2009)
22. Klein, G., Feltovich, P., Bradshaw, J., Woods, D.D.: Common ground and coordination in joint activity. In: Rouse, W., Boff, K. (eds.) Organizational Simulation, pp. 139–184. Wiley, New York (2004)
23. Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., Prasad, M.N., Raja, A., Vincent, R., Zuan, P., Zhang, X.O.: Evolution of the GPGP/TAEMS domain-independent coordination framework. Autonomous Agents and Multi-Agent Systems 9, 87–143 (2004)
24. Mendonca, D., Jefferson, T., Harrald, J.: Emergent interoperability: Collaborative adhocracies and mix and match technologies in emergency management. Communications of the ACM 50(3), 45–49 (2007)
25. Mendonca, D., Wallace, W.A.: A cognitive model of improvisation in emergency management. IEEE Systems, Man and Cybernetics: Part A 37(4), 547–561 (2007)
26. Mintzberg, H.: Structure in 5's: A synthesis of the research on organization design. Management Science 26(3), 322–341 (1980)
27. Mintzberg, H.: Structure in Fives: Designing Effective Organizations. Prentice-Hall, New Jersey (1983)
28. Odell, J., Nodine, M., Levy, R.: A Metamodel for Agents, Roles, and Groups. In: Odell, J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 78–92. Springer, Heidelberg (2005)

29. Odell, J.J., Van Dyke Parunak, H., Fleischer, M.: The Role of Roles in Designing Effective Agent Organizations. In: Garcia, A.F., de Lucena, C.J.P., Zambonelli, F., Omicini, A., Castro, J. (eds.) SELMAS 2002. LNCS, vol. 2603, pp. 27–38. Springer, Heidelberg (2003)
30. Omicini, A., Ricci, A., Viroli, M.: Coordination Artifacts as First-Class Abstractions for MAS Engineering: State of the Research. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 71–90. Springer, Heidelberg (2006)
31. Parunak, H., Brueckner, S., Fleischer, M., Odell, J.: A preliminary taxonomy of multi-agent interactions. In: Proceedings of The Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003), Melbourne, Australia, pp. 1090–1091 (2003)
32. Prue, B., Voshell, M., Woods, D., Peffer, J., Tittle, J., Elm, W.: Synchronised coordination loops: A model for assessing distributed teamwork. In: Proceedings of NATO Research and Technology Organisation (RTO) Human Factors and Medicine Panel (HFM) Symposium. Adaptability in Coalition Teamwork, pp. 17.1–17.12 (April 2008)
33. Rahwan, I., Juan, T., Sterling, L.: Integrating social modelling and agent interaction through goal-oriented analysis. Computer Systems Science and Engineering 3, 87–98 (2006)
34. Sandholm, T., Lesser, V.R.: Coalitions among computationally bounded agents. Artificial Intelligence 94, 99–137 (1997)
35. Scerri, P., Pynadath, D.V., Schurr, N., Farinelli, A., Gandhe, S., Tambe, M.: Team Oriented Programming and Proxy Agents: The Next Generation. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS (LNAI), vol. 3067, pp. 131–148. Springer, Heidelberg (2004)
36. Sims, M., Corkill, D., Lesser, V.: Automated organization design for multi-agent systems. Autonomous Agents and Mult-Agent Systems 16, 151–185 (2008)
37. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
38. Tidhar, G.: Organization-Oriented Systems: Theory and Practice. PhD Thesis, Department of Computer Science, University of Melbourne (1999)
39. van Diggelen, J., Bradshaw, J.M., Johnson, M., Uszok, A., Feltovich, P.J.: Implementing Collective Obligations in Human-Agent Teams Using KAoS Policies. In: Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., da Silva, V.T., Matson, E., Polleres, A. (eds.) COIN 2009. LNCS, vol. 6069, pp. 36–52. Springer, Heidelberg (2010)
40. Weigand, H., Dignum, V., Meyer, J.-J., Dignum, F.: Specification by Refinement and Agreement: Designing Agent Interaction Using Landmarks and Contracts. In: Petta, P., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2002. LNCS (LNAI), vol. 2577, pp. 257–269. Springer, Heidelberg (2003)
41. Yen, J., Fan, X., Sun, S., Hanratty, T., Dumer, J.: Agents with shared mental models for enhancing team decision-makings. Journal of Decision Support Systems: Special Issue on Intelligence and Security Informatics 41(2) (2005)

An Agent-Based Inter-organizational Collaboration Framework: OperA+

Jie Jiang, Virginia Dignum, and Yao-Hua Tan

TPM, TU Delft, The Netherlands
{jie.jiang,m.v.dignum,y.tan}@tudelft.nl

Abstract. Inter-organizational collaboration often occurs in complex, dynamic and unpredictable environments. Regulating structures should be represented explicitly and independently from the acting components at different levels of abstraction in order for stakeholders to be able to analyze the overall setup and decide on their participation. This paper proposes a framework for describing collaboration relationships in inter-organizational partnerships. The framework is based on the OperA model for multi-agent organizations and provides a specification dimension which describes what objectives to achieve from the designer's perspective based on the concept of organization, role and role dependency, and an enactment dimension which represents the agents and their role enacting relationships to describe who achieves what objectives from an implementation perspective. The adoption of composite roles and composite agents facilitates a modeling of nested organizations, which offers a suitable way to manage/regulate inter-organizational interactions.

Keywords: inter-organizational collaboration, multi-agent system, OperA, compositional design.

1 Introduction

Socio-technical systems are complex adaptive entities that require the engagement of social and technical elements in an environment to reach certain goals [14]. Such systems are composed of interconnected components whose interactions form processes at multiple levels of abstraction. Modeling and analysis of such systems is difficult because (1) it is impossible to elaborate everything at a single aggregation level, (2) requirements and functionalities are not fixed a priori, (3) components are not designed nor controlled by a common entity, and (4) unspecified changes may occur during runtime. Examples of such systems are inter-organizational projects, supply chains, behavior in energy markets, and introduction of new policies etc.

Agent-based models have been increasingly developed and adopted to describe, analyze and simulate socio-technical systems and explore phenomena concerning complex relations between entities (e.g., [8], [7]). Comprehensive analysis of agent systems has shown that organization frameworks are needed for the appropriate modeling of complex systems with MAS where many independent

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 58–74, 2012.

entities coexist within explicit normative and organizational structures [1]. The deployment of organizations in dynamic and unpredictable settings brings forth critical issues concerning the design, implementation and validation of their behavior, and should be guided by two principles:

- Provide sufficient representation of the organizational requirements so that the overall system complies with its objectives,
- Provide enough flexibility to accommodate heterogeneous components.

A number of methodologies with a clear organizational vision have already been proposed, such as AGR [15], Roadmap [13], Gaia [17], INGENIAS [16], Moise+ [11]. Nevertheless, current practice of Multi-Agent System (MAS) design tends to take agents as atomic entities [6], which are presented at the same level of abstraction. This either leads to an extremely large model that tries to describe everything at a single aggregation level or a vague model without sufficient information to guide or regulate the actors. Another problem of an extremely large model is the lack of controllability when unspecified changes occur. To solve these problems, the idea of compositionality should be considered, which enables to define and integrate processes and knowledge at different levels of abstraction [3]. The higher levels model the systems in terms of coarser-grained components while the lower levels provide increasing details to the components designed and controlled by different entities. This not only enables integrating different types of components in one model and providing necessary opacity in inter-organizational systems but also makes it easier for actors to understand their partnerships. Thus, components and groups of components can be easily reused at all levels of design. Compositionality also supports open systems since (external) participants can extend existing designs according to their own situations and designers are free of foreseeing details about participants.

Moreover, to enable flexible role enactments given that the implementation of a system does not deviate from the original goals, there is a need for representing the regulating structures explicitly and independently from the acting components. That is, regulations have to be specified in a separate picture to, on the one hand, guide the agents' behavior, and on the other hand, provide enough autonomy for the agents' participation.

Considering the importance of compositionality and the need to explicitly represent the regulating structures independently from the acting components, this paper presents a framework OperA+ which extends the social structure in OperA [10] modeling framework with constructs to represent multi-organizational interactions in two dimensions. The *specification dimension* presents the regulating structures in terms of connected roles and organizations while the *enactment dimension* presents the acting components in terms of agents enacting the roles. The concepts of *composite roles* and *composite agents* facilitate the composite structure of the framework, which provides users with a multi-level modeling environment. The higher-level specification captures the commonalities of organizational collaborations while the lower-level specifications present the individualities by layers upon layers of customization according to more specific requirements. Components at the same level are modeled separately through

lower-level specifications, which decreases their mutual influence when one of them changes. For example, most of the supply chains include exporter, carrier, importer at an abstract level but a food supply chain and a textile supply chain are different in the details of the carrier due to the different requirements of transporting food and textile.

The paper is organized as follows. Section 2 gives a brief introduction of the OperA methodology. Section 3 illustrates the proposed framework OperA+ by presenting its meta-model and formal definitions with an example of an existing EU project. In Section 4, we give some design guidelines. Thereafter, related work is discussed in section 5. Finally, conclusions and directions for future work are presented in Section 6.

2 Background

The OperA framework [10] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims and the agents who act in it. That is, OperA enables the specification of organizational structures, requirements and objectives independently from any knowledge on the properties or architecture of agents, which allows participating agents to have the freedom to act according to their own capabilities and demands. The OperA framework consists of three interrelated models: organization, social, and interaction.

The *Organizational Model* (OM) is the result of the observation and analysis of the domain and describes the desired behavior of the organization, as determined by the organizational stakeholders in terms of roles, objectives, norms, interactions and ontologies. The design and validation of OperA OMs can be done with the OperettA tool [1]. The OM provides the overall organization design that fulfills the stakeholders' requirements. The OM consists of a *social structure* which describes roles and their objectives, and relations between roles concerning achievement of objectives, a *normative structure* which describes norms associated with roles, an *interaction structure* which is an abstract workflow that specifies how objectives should be achieved by the organization using the notions of scenes and landmarks, and a *communicative structure* which specifies the organization's ontology and communicative languages.

The OM is a descriptive view of the organization. In itself, the OM cannot act but is dependent on a population of agents that enact its roles in order to achieve the organization's objectives. What this population looks like and how it acts are described in the Social and Interaction Models in OperA.

The *Social Model* (SM) maps roles to agents and describes agreements concerning the role enactment and their conditions in social contracts. Roles are typically declarative entities meant to represent a part of the organization's design and can be taken up by the agents enacting the role. Objectives of an organization are achieved through the actions of agents, which means that, at each moment, an organization should employ the relevant agents that can make its objectives happen. That is, a role only gets an operational semantics

indirectly through the agents that take up that role. For the operationalization of OperA organizations, a *Gatekeeper* role is defined, which is responsible for the assignment of roles to (external) agents. The gatekeeper agent is part of the SM of each organization.

Finally, the *Interaction Model* (IM) specifies the interaction agreements between role-enacting agents as interaction contracts. The IM specification enables variations to the enactment of interactions between role-enacting agents.

In this paper, OperA+ mainly concentrates on the social structure of OM and extends the OperA framework by providing compositionality to both roles and agents in two dimensions, which enables describing and analyzing organizational interactions at multiple levels of abstraction. Thus, a balance can be achieved between conformity and autonomy from the perspectives of both designers and actors.

3 The Proposed Framework

3.1 Fundamental Concepts

OperA+ describes two dimensions of analysis:

- the **specification dimension** which describes what objectives to achieve in the inter-organizational collaboration, and
- the **enactment dimension** which describes who achieves what objectives based on the specification dimension.

The metamodel of the OperA+ framework in Figure 1 shows the main concepts and their relationships. Note that this metamodel mainly concentrates on the social structure in OperA and is a part of the complete OperA+ framework for modeling multi-organizational collaborations. The extensions of other structures such as the normative structure [12] and interaction structure are explored in other works.

A specification of an inter-organizational collaboration model starts from a role and continues with a set of organizations at different levels of abstraction. An organization is a social arrangement which pursues collective objectives through a set of connected roles. Roles are typically declarative entities meant to represent a part of the organization's design. In OperA+, we refine roles into two kinds: atomic role and composite role. Each composite role refers to a unique organization at a lower level in the hierarchy which elaborates the objectives of the composite role into finer-grained roles and gives more constraints or information on how to accomplish the objectives. Atomic roles are not further specified enabling heterogeneous enactment.

Definition 1 (role). A role r is a tuple $(obj_r, RCap_r, org_r)$ such that*:*

- obj_r is a set of objectives*,*
- $RCap_r$ is a set of capabilities required by the role to accomplish the objectives*,*

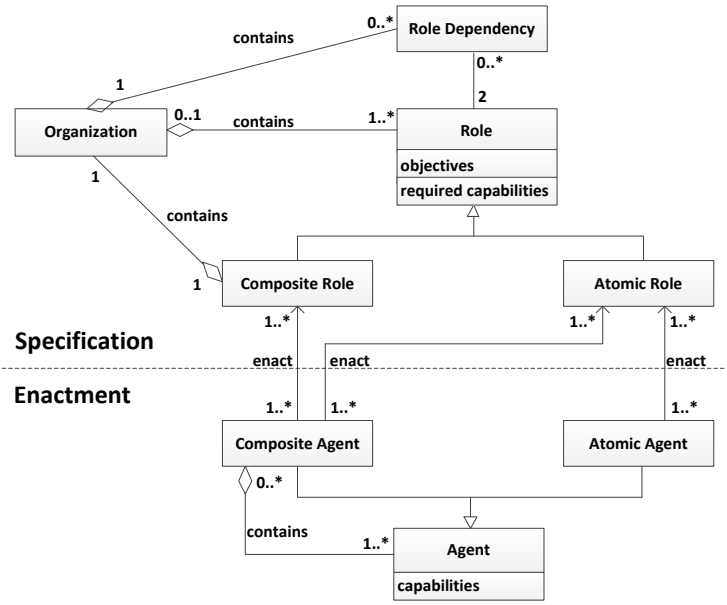


Fig. 1. Meta-model of the OperA+ framework

- $org_r = (R, Dep)$ is an organization which contains a set of roles $R = \{r_1, r_2, \cdots, r_n\}$ and a set of role dependencies $Dep = \{dep_1, dep_2, \cdots, dep_m\}$ where $dep = (r_i, r_j, obj)$, $r_i, r_j \in R$, $obj \in Obj_{r_i} \bigcup Obj_{r_j}$.

Objectives indicate the desired results a system envisions. Each role has a set of objectives that it seeks to achieve. For this purpose, certain capabilities are required for the agents who want to play a specific role. Capabilities are used to define a skill or capacity for achieving the objectives. Objectives and capabilities are expressed as predicates. When $org = \phi$, r is an atomic role which is not further elaborated. When $org \neq \phi$, r is a composite role which is elaborated as a set of roles R, and their dependencies Dep through which objectives can be passed unidirectionally. The dependency relation between roles r_i and r_j for objective obj, represented as (r_i, r_j, obj), indicates that obj can be passed from r_i to r_j, i.e., r_j can realize obj for r_i. In this paper, we overload all the definitions for simplicity, e.g., r is used as a reference to the definition $r = (Obj_r, RCap_r, org_r)$. Besides, we use subscripts to indicate the elements in each definition, e.g., Obj_r, $RCap_r$, and org_r refer to the objectives, required capabilities and organization of r.

To illustrate our proposal, we use as running example the organization of an existing EU project that is being undertaken by a large number of universities, companies, interest groups and industry associations. The project itself has the nature of research collaboration and is seen as a role with the objective of

creating an inventive way of enabling a faster, safer and more reliable international governance structure in a certain domain. Based on the role template in common research collaborations, an organization of three interdependent roles are identified at an abstract level according to the requirements of the project, which are illustrated in Table 1. The three roles together describe the top level organization and collaboratively realize the objective of the project. It can be seen that the objectives of the three roles are only intellectual attitudes of what should be done in the project, which do not contain detailed information on how to realize the objectives.

Table 1. Role table for the project organization

Role	*Role Objective*	*Required Capability*
Project Manager	Overall administration and coordination	Leadership
Research & Development	Create reliable, secure, and cost effective logistic chains supporting all applicable regulations and procedures	Domain knowledge, research and development experience
Knowledge Disseminator	Valorization and dissemination of the research results among potential users	Understanding and explaining abilities

In order to accomplish its objectives, each role in the organization must coordinate several tasks. As a result, the three roles are elaborated to three lower-level organizations shown in Table 2. In each lower-level organization, more specific roles are specified to give more information or constraints on how to accomplish the objectives of the corresponding composite role.

In the enactment dimension, roles in a specification are enacted by agents to realize the objectives. Since our model is targeting at open systems in which agents are not known at design time, the enactment dimension is only an illustration of the possible solutions for the specification dimension. A specification can have multiple enactments at different times and circumstances, i.e., the enacting of a specification is evolving with the changes of the agents. In OperA+, an enactment of a specification is defined as a set of agents and their mappings to the roles. Agents are autonomous, socio-cognitive entities capable of social behavior. Within inter-organizational collaborations, agents must have the ability to communicate with each other, and work to achieve the objectives of the roles that match their capabilities and at the same time they accept to play. Two kinds of agents are defined in OperA+: atomic agent and composite agent.

Definition 2 (agent). An agent a is a tuple (Cap_a, A_a) such that*:*

- Cap_a is a set of capabilities*,*
- $A_a = \{a_1, a_2, \cdots, a_k\}$ is a set of agents*.*

Table 2. Role table for the lower-level organizations

Organization	*Role*	*Role objective*
Project Manager	Project Director	Overall legal, contractual, ethical, financial administration
	Project Coordinator	Maintain communication with each partner
Research & Development	Service Architecture Builder	Develop solutions for service architecture and explore their potential of process innovation
	Information Governance Modeler	Design an information governance model to ensure that information is available where and when needed
	Semantic Modeler	Develop a methodology for modeling semantics
	Concept Proofer	Prove the technical feasibility of the concepts
Knowledge Disseminator	ICT Disseminator	Disseminate knowledge among commercial service providers
	Logistic Disseminator	Disseminate knowledge among logistic service providers

In order to enact a particular role, an agent must possess a sufficient set of capabilities that allow the agent to carry out the role and achieve its assigned objectives. Therefore, corresponding to the required capabilities of roles, a key component of agents is the capabilities they actually possess. Since agents in an organization as well as their individual capabilities may change over time, the ability of the agent to play specific roles also changes, which may result in different sets of enactments for a specification.

When $A_a = \phi$, a is an atomic agent. An atomic agent is an atomic entity whose inner structure is either invisible or unimportant to the other parts of the system. It can be an individual, a service, or even a company whose inner world is hidden from the outside.

When $A_a \neq \phi$, a is a composite agent. A composite agent is a group of agents who usually share a set of objectives and follow the same set of norms. Each sub-agent in a composite agent can be either atomic or composite. The capabilities of a composite agent is the union of its sub-agents: $Cap = \bigcup_{a' \in A_a} Cap_{a'}$. For example, a university which includes a group of faculties, teachers, students, etc., is a composite agent.

3.2 Types of Role Enactment

In the proposed framework, the following rules are applied to role enactments. From the perspective of roles, an atomic role can be enacted by any type of agents while a composite role can be (1) directly enacted by a composite agent providing that the internal organization of the agent matches that of the composite role, or (2) indirectly enacted by a set of independent agents each enacting a sub-role

respectively. From the perspective of agents, each agent can enact one or more roles if its capabilities meet the requirements of the roles. Moreover, when an agent enacts a role, it can further extend the role specification according to its own requirements and functionalities which are not known in advance.

It should be noticed that each role can be enacted by a single agent or by multiple independent agents of the same kind. When a role is enacted by multiple agents of the same kind, it means that the agents accomplish the objectives multiple times independently. For example, several people independently enact the role of cashier in a supermarket.

A specification can have multiple enactments. For one enactment, there are a set of agents enacting the roles in the specification. Some of the agents may have their own understanding of the objectives of the roles they enact. Therefore, agents can extend the inherited specification according to their own capabilities, i.e., they may further refine the specification to better achieve the objectives. In this way, a balance between conformity and autonomy can be achieved.

3.3 Towards a Formal Framework for OperA+

The specification dimension of OperA+ is from the designer's perspective. It describes organizational collaborations in terms of interrelated objectives that should be achieved, i.e., what roles are needed and how they interact with each other. Business rules and social norms applying to this collaboration can be described by elaborating on the inner structure of the roles at different levels of abstraction and by the use of normative models. In this paper, we focus on the structural aspects and leave normative issues for future work.

Definition 3 (specification). A specification S of an inter-organizational collaboration model is a role r_{0S}.

A role in our model is a tree-like structure consisting of sub-roles and role dependencies at different abstraction levels. That is, a specification of an inter-organizational collaboration model is a set of hierarchically organized entities and their collaborating relations. To simplify the following definitions based on specifications (i.e. r_{0S}), we define the function *unbox*. Given a role r, *unbox* returns all the roles in the specification starting from r.

Definition 4 (unbox). Let r be a role. We define the function *unbox* as follows: $unbox(r) = R_{org_r} \bigcup (\bigcup_{r' \in R_{org_r}} unbox(r'))$.

Then the set of all roles in a specification starting from r can be obtained. As a role is either atomic or composite, the set of all roles is divided into two subsets, i.e., the set of all atomic roles and the set of all composite roles. Given a role r, we can obtain the two subsets respectively by the following functions.

$R^*_A(r) = \{r' | r' \in unbox(r) \wedge org_{r'} = \phi\}$,
$R^*_C(r) = \{r' | r' \in unbox(r) \wedge org_{r'} \neq \phi\}$.

To accomplish the objectives of the roles following tree-like structures, a description from an implementation perspective (i.e., an enactment) is required.

From a practical perspective, an enactment can be seen as a set of arrangements of objectives in collaborations. We define an enactment as follows.

Definition 5 (enactment). An enactment E of a specification S is a tuple (S, A^*, RE) such that*:*

- A^* is a set of agents such that $A^* = A^*_A \bigcup A^*_C$, $A^*_A \bigcap A^*_C = \phi$, where A^*_A is the set of all atomic agents and A^*_C is the set of all composite agents,
- $RE = \{re_1, re_2, \cdots, re_d\}$ is a set of role enactments where $re = (r, a), r \in unbox(r_{0S}), a \in A^*$,
- $\forall (r, a) \in RE : RCap_r \subseteq Cap_a$,
- $\forall (r, a) \in RE, r \in R^*_C(r_{0S}), \exists a \in A^*_C$*:* $(\forall r' \in \{r'' | r'' \in org_r, org_{r''} = \phi\}, \exists a' \in A_a : (r', a') \in RE)$.

A^* specifies all the agents participating in an enactment. RE is used to indicate which agent enacts which role. For the moment, we assume a role-enacting agent to be such that $\forall (r, a) \in RE : RCap_r \subseteq Cap_a$, i.e., when an agent enacts a role, its obtained capabilities cover the required capabilities of the role. $\forall (r, a) \in RE, r \in R^*_C(r_{0S}), \exists a \in A^*_C$: $(\forall r' \in \{r'' | r'' \in org_r, org_{r''} = \phi\}, \exists a' \in A_a : (r', a') \in RE)$ indicates that if a composite agent enacts a composite role, it must contain corresponding sub-agents for all the atomic roles in the composite role.

It can be seen that Definition 5 only describes the formation rules of an enactment but not specify whether a specification can be fulfilled by the enactment. Therefore, we give the definition of complete enactment as follows.

Definition 6 (complete enactment). An enactment $E = (S, A^*, RE)$ is a complete enactment when $\forall r \in R^*_A(r_{0S}), \exists a \in A^* : (r, a) \in RE$*,* indicating that for each atomic role in the specification S*,* there is at least one agent enacting it.

In open inter-organizational collaborations where agents are free to decide their partnership, an enactment may not be a complete enactment. While in closed inter-organizational collaborations, an enactment must be a complete enactment to make sure that the objectives of corresponding specifications can be accomplished.

In our example, a particular instance of specification and enactment is illustrated in Figure 2. In the project, the *Project Manager* in the top level organization is enacted by a research and technology organization in which two employees enact the sub-roles *Project Director* and *Project Coordinator*. The *Knowledge Disseminator* is enacted by two independent agents. One is an academy on logistics while the other is a set of interest groups. The *Research & Development* is enacted through four independent agents each enacting a sub-role. In particular, the *Information Governance Modeler* is enacted by a university that transforms this atomic role into a composite role and further elaborates it into an organization containing two atomic roles *Team leader* and *Team member* shown at the

lowest level. The *Team leader* is enacted by a person while the *Team member* is multi-enacted by a group of researchers. All the role enacting agents in this project are selected by the project owner through comparing the capabilities of the candidate agents with the required capabilities of the roles. In addition, if a new role *Norm modeler* is added in the organization of *Research & Development*, it will only influence the related roles and their enacting agents within this organization, which makes the whole model more stable. It can be seen that the organizational interactions of the project are described from abstract to concrete in a hierarchical structure, which not only decomposes the complexity of the project into sub components that can be designed or controlled by different entities, but also provides flexile role enactments for agents since they can further elaborate the role specification according to their own characteristics.

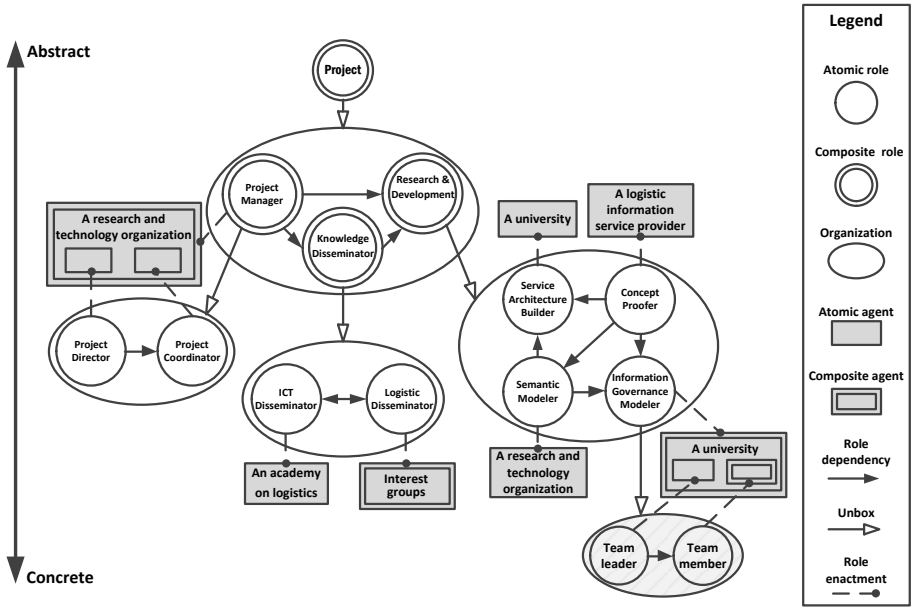


Fig. 2. The specification and enactment of the project

4 Design Guidelines

OperA+ describes organizational interactions at multiple levels of abstraction with the process of role elaboration in terms of composite roles and organizations. After a number of case analysis using the OperA+ framework in our research, we give the following guidelines.

4.1 Role Identification

Roles are the core building blocks of organizational collaboration models. They give instructions for agents' interactions that will finally achieve the objectives of a system. An organization is a set of interdependent roles. Therefore, the global behavior embodied by an organization is decomposed into finer-grained behaviors captured by the roles in the organization. When identifying roles in an organization, the following aspects can be considered.

(1) Functions. An organization is designed for a specific purpose which defines the overall function of the organization. Role identification in an organization results in the distribution of the overall function to a set of smaller functions. Therefore, when a relatively independent sub function can be derived from the overall function of a system, the sub function may indicate a role in the organization of the system. For example, in an international trade organization, the overall function is to transport goods from the place where the goods are produced to the place where the goods are demanded. Within this organization, the overall function can be decomposed into several sub functions such as producing, exporting, importing, which can be accomplished by different entities. Correspondingly, we will identify the roles as producer, exporter, importer.

(2) Regulations. In addition to the functions, regulations are another important issue that need to be taken into account for role identification. In real applications which involve authorities as a main participant, regulations are always a key element for role identifications. For example, in international trade, customs is a very important role which regulate other participants' behavior. Therefore, customs can be seen as a focal point of the organization from which other roles can be identified through the regulative relationships they have with the customs.

4.2 Lower-Level Organization Identification

A lower-level organization is an elaboration of a composite role in terms of finer-grained sub-roles which contain more specific objectives and dependencies. However, if the specification is too detailed, i.e., the specification contains a lot of constraints describing by lower-level organization of roles, the agents enact the specification will have little autonomy to achieve the overall objectives, which is not the case in socio-technical systems. On the other hand, if the specification is very abstract, the agents will not have sufficient guidance and constraints to perform the tasks, which may lead to deviation from the original goals of the system. Therefore, a balance should be achieved when designing lower-level organizations.

(1) Importance in the Regulative Process. In an organization, some of the roles are more important with respects to their functionalities and their connections with other roles. These roles normally need more analysis and regulations.

For example, exporters and importers are the main roles in international trade since they have more interactions with other participants such as producers, logistic service provider, customs. Such roles generally need to be specified at lower levels for clearness of their organization so that customs can regulate them more easily and other participants can have better interactions with them. On the other hand, logistic service provider, whose main objective is transporting goods from one place to another in time, is usually considered as a supporting role to help exporters/importers achieve their objectives. Thus, we do not need to elaborate its organization in most of the cases. Besides, customs is a very important role since it regulates the whole business process and functions as the director in international trade.

(2) Capabilities. Capabilities are the key factor to determine which agents can be assigned to which roles within an organization. If the required capabilities of a role are quite diverse and related to several domains, the role usually needs to be elaborated in a lower-level organization in which capabilities of the same domain are assigned to a sub-role. For example, in the EU project we introduced in Section 3, the role of Research & Development requires knowledge of several domains such as service architecture, information governance. In order to employ agents who have expertise in these different domains, the role is refined into a lower-level organization which contains more detailed explanation of what objectives should be done and what kind of domain knowledge is required.

4.3 Modeling Process

Following a modular way, an OperA+ model subdivides a system into lower-level organizations that can be independently designed and then re-used in different systems to drive multiple functionalities. Based on the compositionality of roles and agents in OperA+, we divide the whole modeling process into four phases as shown in Figure 3. The vertical dimension is from abstract contexts to specific contexts while the horizontal dimension is from roles to agents. In abstract contexts, organizational interactions are described by roles with very general objectives so that agents have much autonomy but little guidance and constraints. To ensure that the objectives of the roles can be accomplished, on the one hand, roles in abstract contexts will be elaborated with more specific information, and on the other hand, eligible agents will be admitted to join in the organization and enact the roles. Finally, the whole picture of the organizational interaction model can be obtained.

(1) Roles in Abstract Contexts. For a system to be modeled, there must be some initiative objectives that trigger the system. These objectives are always described at an abstract level and only give the general ideas about what to do. The roles in this phase can be derived from related applications according to

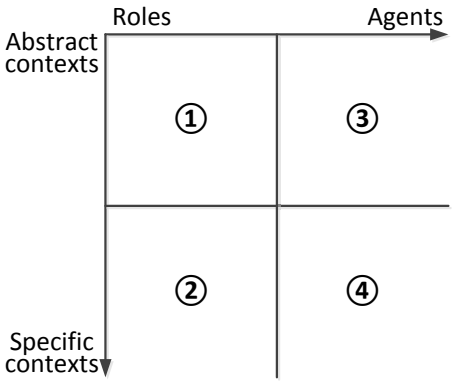


Fig. 3. Modeling process

the initial requirements of the model. Therefore, they can be reused by other organization models with similar requirements. In practice, organizational collaboration models are of different types such as inter-organizational projects, supply chains. These types of collaboration models have their own universal set of basic roles which can be seen as templates. Based on the nature of an organizational collaboration, we can inherit the basic roles from the related templates. That is, we have some patterns to use in the first phase. Specifically, some organizational collaboration models may cover multiple types and they can inherit the basic roles from all the templates of these types. For role dependencies, a similar mechanism can be used.

(2) Roles in Specific Contexts. Based on the roles derived from abstract contexts, further elaboration is needed for specific applications. In this phase, the roles are case specific and are obtained based on the characteristics of specific collaboration requirements. Organizational collaboration models with the same set of roles in an abstract context may have different requirements in specific contexts and result in different lower-level organizations of sub-roles. That is, with the refinement of contexts, an organizational collaboration model at an abstract level is contextualized into finer-grained organizations of components with respect to different application environments. With more specific objectives, agents can know what is actually needed to be done in a specific situation. However, "specific" is in a relative sense and the degree of specificity should balance both indication and autonomy. Therefore, on the one hand, enough regulations are imposed on the agents' behavior so that the implementation of the system does not deviate from the original goals. On the other hand, the agents have enough autonomy to decide their participation and collaborating strategies.

(3) Agents in Abstract Contexts. In accordance with the roles in abstract contexts, potential agents can be identified. Generally speaking, no single agent can accomplish the objectives of a role in an abstract context. This is due to the fact that organizations in real business world usually focus on a few business fields and have limited resources. No single organization can do all the work in a large business process that involves several business fields and requires a large quantity of resources. Therefore, we should construct a pool of potential enactors, i.e. candidate agents. All the qualified agents can be added into the pool for further selection. In this phase, we only know the general objectives from the roles in abstract contexts, and these objectives serve as the basic requirements for the agents who want to participate in an organizational collaboration.

(4) Agents in Specific Contexts. In the last phase, we should figure out the specific agents for the roles in a specific context. Then the objectives of the roles can be accomplished as expected. For this, the second phase and the third phase which describe the different aspects of the collaboration model should be combined to derive the final results. According to the roles in a specific context, we know what objectives to achieve in a specific situation. From the agents in an abstract context, we select appropriate agents for the roles in specific contexts. Finally, we know exactly who achieves what objectives and the whole model is constructed.

5 Related Work

Agent-based systems are increasing both in size and diversity. This growth is pushing agent-based systems beyond a size that is manageable by individual organizations [15]. Thus, there is a growing need for the use of organization frameworks in agent societies. One of the first models of agent organizations is the AGR model [15]. An AGR model describes an organization as a role-group structure imposed on agents, which provides the basic foundational elements required in multi-agent systems to foster dynamic group formation and operation. In AGR, a group is used as a container to organize roles and agents but it does not consider roles and agents themselves as composite entities which can be represented at multiple abstraction levels.

DESIRE [2] is a compositional framework for modeling multi-agent tasks, which focuses on the composite tasks of a single agent not the composite structure of the agent itself. Moise+ [11] is an organizational model that considers the structure, the functioning, and the deontic relation among them. In Moise+, three main concepts, roles, role relations, and groups, are used to build the individual, social, and collective structural levels of organization, but agents are not further refined as structural components. MaSE methodology [9] is a full-lifecycle methodology for analyzing, designing, and developing heterogeneous MASs. Roles are transformed from the goal hierarchy diagram and form the

foundation for agent classes and correspond to system goals during the design phase, while agents are defined in terms of the roles they will play and the conversations in which they may participate.

Extended Gaia [17] is an agent-oriented methodology based on the organizational concepts of roles, interactions, and organizations, but neither roles nor agents are considered as composite components that are modeled at multiple levels of abstraction. Roadmap [13] is another extension to Gaia, which has four improvements: formal models of knowledge and the environment, role hierarchies, explicit representation of social structures and relationships, and incorporation of dynamic changes. Tropos [4] adopts the notions of actor, goal, plan, etc., in all phases of software development, but an actor is used to represent a physical, social or software agent as well as a role or position, which does not explicitly represent the regulating structures independently from the acting components.

INGENIAS [16] provides five meta-models (organization, environment, tasks/goals, agent and interaction) to guide the development of a MAS in which the agent model describes individual agents without composite structures. In [5], a formal role-based framework is proposed for modeling organizations, which only explores the composite structure of roles while takes agents as black boxes. Based on a holonic organizational metamodel, ASPECS [6] provides a suite of refinement methods for modeling systems at different levels of details from requirements to code. The vision of holons is similar with the recursive and composed agent in MASs while our proposal explores the composite structures of both roles/organizations and agents from two dimensions and establishes a flexible combination.

6 Conclusion and Future Work

This paper proposes the framework OperA+ to model inter-organizational collaborations at different levels of abstraction. OperA+ presents a full illustration of *who* achieves *what* objectives in two dimensions of specification and enactment with initial steps of formalization. In particular, it provides a flexible way of modeling organizational interactions by which designers and actors can achieve a balance between conformity and autonomy. Targeting at modeling complex systems where autonomous entities interact with each other to achieve collective goals and those entities again have inner structures in which a set of sub-entities coexist, OperA+ facilitates a composite way of modeling organizational interactions, which not only distributes the complexity of the whole system but also provides a mechanism to show the commonalities and individualities of an inter-organizational collaboration model from different perspectives. In OperA+, formal specifications at an abstract level can have different extensions according to different situations. To help users build their own applications using OperA+, some design guidelines are provided.

In future work, we intend to design an algorithm for the mappings between roles and agents according to their properties. That is, how to find a proper

enactment for a specification of an organizational model to better achieve the objectives. Currently, the OperA+ framework only concentrates on the social structure of the organizational model in OperA. In order to build an integral agent society supporting inter-organizational collaborations, other structures and models in OperA also need to be elaborated. Moreover, we will develop a simulation platform to evaluate different extensions of an abstract specification based on a set of indicators.

References

1. Aldewereld, H., Dignum, V.: OperettA: Organization-Oriented Development Environment. In: Dastani, M., El Fallah Seghrouchni, A., Hübner, J., Leite, J. (eds.) LADS 2010. LNCS, vol. 6822, pp. 1–18. Springer, Heidelberg (2011)
2. Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J.: Desire: Modelling multi-agent systems in a compositional formal framework. International Journal of Cooperative Information Systems 6, 67–94 (1997)
3. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of compositional multi-agent system development. In: Proceedings of the 15th IFIP World Computer Congress, WCC 1998, Conference on Information Technology and Knowledge Systems, IT and KNOWS 1998 (1998)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Journal of Autonomous Agents and Multi-Agent Systems 8, 203–236 (2004)
5. van den Broek, E.L., Jonker, C.M., Sharpanskykh, A., Treur, J., Yolum, P.: Formal Modeling and Analysis of Organizations. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM 2005 and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 18–34. Springer, Heidelberg (2006)
6. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. Journal of Autonomous Agents and Multi-Agent Systems 20, 260–304 (2010)
7. Dalpiaz, F., Ali, R., Asnar, Y., Bryl, V., Giorgini, P.: Applying tropos to socio-technical system design and runtime configuration. In: Proceedings of the 9th Workshop on Objects and Agents, WOA 2008 (2008)
8. Dam, K.H.V.: Capturing socio-technical systems with agent-based modelling. TU Delft, PhD Thesis (2009)
9. DeLoach, S.A.: The mase methodology. In: Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook (2004)
10. Dignum, M.V.: A model for organizational interaction: based on agents, founded in logic. Utrecht University, PhD Thesis (2004)
11. Hubner, J.F., Sichman, J.S., Boissier, O.: Moise+: towards a structural, functional, and deontic model for mas organization. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002 (2002)
12. Jiang, J., Aldewereld, H., Dignum, M.V., Tan, Y.: A context-aware normative structure in mas. In: Proceedings of AAMAS (2012)
13. Juan, T., Pierce, A., Sterling, L.: Roadmap: Extending the gaia methodology for complex open systems. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002 (2002)

14. Nikolic, I.: Co-Evolutionary Method For Modelling Large Scale Socio-Technical Systems Evolution. TU Delft, PhD Thesis (2009)
15. Odell, J., Nodine, M., Levy, R.: A Metamodel for Agents, Roles, and Groups. In: Odell, J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 78–92. Springer, Heidelberg (2005)
16. Pavón, J., Gómez-Sanz, J.J., Fuentes, R.: The ingenias methodology and tools. In: Agent-oriented Methodologies (2005)
17. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Multi-agent systems as computational organizations: The gaia methodology. In: Agent-oriented Methodologies (2005)

MANET: A Model for First-Class Electronic Institutions

Charalampos Tampitsikas[1,2], Stefano Bromuri[2], and Michael Ignaz Schumacher[2]

[1] Università della Svizzera italiana,
Communication Sciences Department,
via G. Buffi 13, 6900 Lugano, Switzerland
charalampos.tampitsikas@usi.ch
[2] University of Applied Sciences Western Switzerland,
Institute of Business Information Systems, 3960 Sierre, Switzerland
{stefano.bromuri,michael.schumacher}@hevs.ch

Abstract. In this paper we present a new approach to model electronic institutions (EIs) that are situated in agent environments where heterogeneous agents reside. An EI is seen here as an entity that is deployed within an environment infrastructure that directly mediates the agents' interaction. The environment allows the rules of the EIs, in terms of powers, permissions and obligations to be perceivable as first class entities by the agents belonging to the institution. We express EIs as first class abstractions that can be inspected, manipulated and modified, created and destroyed by the agents populating the agent environment where the institution resides. To represent the EIs we utilize the Object Event Calculus (OEC) formalism that deals with the evolution of complex structures in time and we extend it to deal with the mediation of the events and with the perception of complex structures and events within institutions. We use an e-Health marketplace scenario based on Dutch auctions to illustrate the properties of our model.

Keywords: multi-agent systems, normative systems, electronic institutions, agent environments, logic programming.

1 Introduction

In the Web 3.0 [16], human beings and software applications freely interact to carry out complex activities, inclusive of (but not limited to) e-business and e-government applications. People and organizations delegate many of their tasks to software applications, called agents. An agent is considered an autonomous entity which observes and acts upon an environment and directs its activity towards achieving its goals [30]. These agents behave as representatives acting both reactively and proactively in their principal's interest while they are also empowered to carry out tasks that have legal effects, like signing contracts and performing business transactions.

Many Web 3.0 applications can be defined as complex *open multi-agent systems* (MASs) [15]. A MAS can be considered open [20] when it satisfies the following properties: i) agents are free to join and leave at any time and ii) agents are designed by and represent different stakeholders with different objectives.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 75–92, 2012.

Due to the properties of open MASs, a set of issues must be considered [4]: an open MAS is by definition dynamic as the agents may join or leave at any time; it is insecure as an agent may be programmed to be malicious; it is not deterministic as no agent can have a global knowledge of the system; and finally it has not a central authority. Normative systems, or as Ägotnes et al. specify in [1], systems were social rules apply, try to tackle these issues by defining rules to coordinate heterogeneous agents.

Electronic institutions (EI) [14] are an approach to normative systems, containing a constitutive and regulative part [5]. We can regard an EI as a means for imposing a well-defined structure to the social reality within which agents interact [23], based on a set of rules that mediate the interaction taking place between the agents.

However, EIs suffer from a number of drawbacks. First of all, despite the fact that normative systems provide a level of abstraction in terms of social rules amongst agent societies, it is not clear how these rules mediate the interaction in a MAS in terms of concrete mechanisms. Moreover, the governor agent approach suggested by some enforcement based normative systems [12] has the disadvantage of mixing the concept of infrastructure with the concept of agent, implying that everything in the system is represented as a communicating agent, even when encapsulating low level reactive resources, resulting in quite computationally expensive applications. An example of a model where everything inside the MAS is considered as a communicating agent is the framework proposed by Campos et al. [8] as an extension to electronic institutions.

Secondly, EIs lack of mechanisms that allow the perception of institutional entities and events. The perception of an EI could allow the agents to decide whether participating in the institution would benefit the accomplishment of their goals. Thirdly, current research on normative systems is mainly focused on the communication events inside the system. While communication events are of great importance, they are not sufficient to describe all the possible interactions of the institutional entities and they cannot fully describe the evolution of the system.

To avoid the three drawbacks described above, we present a meta-model which considers the notion of EI as the social constitutive element of an agent environment. In particular, our contribution is to provide a model that proposes the following solutions to the current drawbacks of EIs: i) we introduce the concept of institutional space as the mediator of the social interaction between agents in the agent environment; ii) we propose a perception model to observe institutional spaces and their norms; iii) we present an event system to handle the evolution of institutional spaces and of MASs related to these institutional spaces. We illustrate the perception properties of these concepts by means of an e-Health marketplace example. Although the perception of norms implies that agents can interpret them, for the purposes of this paper we focus only on the social interactions of the agents.

The remainder of this paper is structured as follows: Section 2 is a description of the main properties of agent environment that we have to take into consideration to define our EI meta-model; Section 3 presents our meta-model of first-class Electronic Institutions; Section 4 shows how we apply our model within an e-Health market place based on Dutch auctions; Section 5 shows sketches of our in Prolog; Section 6 puts our work in comparison with existing EI frameworks; finally Section 7 concludes this paper and shows some possible future work directions.

2 Normative Systems and First-Class Agent Environments

Although there is not a clear definition of the agent environment in the traditional normative system models, Esteva in [11] was the first to mention that EIs can be considered as an effort to shape the environment where the agents are situated, offering the agents the conditions to exist and interact. Usually, in the literature, the environment is considered as a domain-specific infrastructure for agents while its main responsibility is the objective coordination of the agents [29].

The agent environment can be used as a first-class abstraction that mediates the interaction between agents taking part in a distributed MAS. First-class abstraction means that the environment is an independent component inside the MAS structure that has its own responsibilities irrelevant to the goals of the agents. According to Weyns [29], the agent environment as a first-class entity can offer four different levels of support:

Basic level: at this level the environment enables agents to access to the *deployment context*. By deployment context, it is meant the external resources with which the MAS interacts (e.g. printers, databases and Web services).

Abstraction level: at this level the environment shields low-level details of external resources defining a standard interface that the agents can access from the environment.

Interaction-mediation level: the interaction mediation level offers support to regulate the access to resources and to mediate the interaction between agents.

Reflection level: The environment supports the modification of its composition and function during runtime. The agents can perceive the properties of the environment and interact in order to modify its state.

A distributed implementation of the model of agent environment proposed by Weyns is represented by the GOLEM agent platform [7]. One of the drawbacks of GOLEM is that it does not model the social interaction amongst the agents, handling it in an ad-hoc manner according to the application, limiting the reusability of the agents and of the infrastructure. Based on the basic level of support proposed by GOLEM, we want to use the abstraction and mediation level of support provided by the agent environment to offer a solution to the main drawbacks of electronic institutions as presented in the previous section and to provide a reusable social interaction model for agent environments. For the purposes of this paper, we study only the environment perception provided by the reflection level and we do not consider run-time modification of its laws.

In order to embed these levels of support into electronic institutions, we first need to specify the appropriate type of normative systems we will use for the mediation of agent interactions. There are two approaches to define normative systems [19]: a)*regimentation based normative systems*, in which a set of rules and protocols are defined to coordinate the behavior of the agent; b) *enforcement based normative systems*, in which some of the agents in the open MAS have the role of regulator agents enforcing the rules when they discover they have been violated.

Regimentation based normative systems are less flexible as it is necessary to specify the rules at design time and the agents are not free to perform actions outside the rules defined by the normative system. The enforcement based approach allows agents to take actions outside the rules of the normative system, but it has the drawback that

sometimes the agents can behave maliciously and not being caught by the enforcer of the law. While proper coordination of agents is crucial, at the same time, it is important to ensure the autonomy of agents. Thus, we envisage electronic institutions as enforcement based normative systems where the agents are free to perform actions outside the rules of the system but for every forbidden action, the system is tracking the violation and applies a corresponding sanction to punish the agent and to preserve its stability. We use enforcement based normative systems that can be created at runtime, but where their rules are first class citizens [25], meaning that the agents can observe the rules.

Moreover, EIs include two basic types of norms: i) constitutive norms and ii) regulative norms. Constitutive norms are based on the notion that "X count-as Y in context C" and are used to support regulative norms by introducing institutional facts in the representation of legal reality [5]. Regulative norms are the main mediation drivers in EIs and are realized by using three main concepts (adapted from [3]) for mediation: power, obligation and permission.

Power specifies that an agent can perform a designated action in a context, which creates or changes an institutional fact. Obligation expresses the idea that at a given time the agent should produce an action as specified by the rules of the normative system. Obligation implies also the concept of prohibition or negative obligation as an action that is forbidden by the rules of the system at a certain time. The concept of permission is both related to the state of the EI and to the concept of power. An agent could either exercise its power, if and only if the institutional conditions permit it (conditional power [13]) or exercise its power even if it does not have the permission to do it. On the second case the agent will be sanctioned by the system. Which of the two previous approaches will be followed depends only on the choice of the EI designer.

3 Modeling First-Class Electronic Institutions

3.1 The MANET Meta-model

The MANET (Multi-agent Normative EnvironmenTs) meta-model is based on the assumption that the agent environment is composed by two fundamental building blocks; the physical environment, concerned with agent interaction with physical resources and with the MAS infrastructure, and the social environment, concerned with the social interactions of the agents and coinciding with the notion of electronic institutions.

In the MANET meta-model we assume that EIs can be composed of three structural components inspired by Stratulat et al. [26]: agents, objects and spaces.

The notion of agent describes the proactive entities within the normative system. For the agents, we assume a separation between a cognitive mind and a physical body with sensors and effectors as described in [7]. The cognitive mind analyzes and reasons about the data received by the sensors as well as reasoning about the agent strategy. The agent uses its effectors to act inside the environment.

The notion of object describes first-class entities that represent virtual entities, virtualizations of external resources or web services, offering an abstraction that hides the low level details from the agents. From the standpoint of EIs, these virtual entities can depict either physical objects either institutional objects.

On one hand physical objects are considered the physical entities of the application (web services, databases, external files etc.) that are present inside an EI. On the other hand, institutional objects, are objects existing only in common agreement amongst the agents of an EI. Institutional objects can be further categorized as: a) objects that can exist within the communication amongst the agents, such as the goods to trade in a market; b) objects that represent agreements between one or more parties; c) objects that represent sanctions for the incorrect behavior of the agent in the EI; d) objects that represent norms of an EI; e) objects that represent institutional spaces, f) and finally objects that represent roles of agents within an EI. Moreover, physical objects can be considered as institutional ones when they obtain institutional attributes during the evolution of the agent environment.

Finally the third structural component of our model are spaces. By default in our model, there always exists a root space which contains all the physical laws (derived from the infrastructure of the MAS) of the system (as first-class objects) and where all the other spaces of the system are been created. But in direct analogy to the human reality where we can distinguish between the physical world and the social world, in MASs we can consider institutional spaces [26] describing the EIs in a normative system. All the institutional spaces of a MASs are situated inside the root physical space.

Institutional spaces constitute a first-class representation of the boundaries and the structure of legal entities like EIs. These spaces include the objects and the agents participating in an EI, and contain information about institutions' topology and configuration. The term boundary here implies that spaces specify the limits of the effects of the events performed by the agents. In our model we suppose that the effects of an event produced inside one space hold only for that space. Since spaces are the boundaries and containers of events, they manage norm violations and fulfillments. The content of each event and the combination of role/power of the agents that produced the event are always checked by the space against the corresponding norms. In case of a norm violation, a space will retrieve the information of the appropriate sanction objects and will apply them to the agent that did not comply with the rules of the system. In other words, we see institutional spaces as structures whose state exist in the physical environment, that is perceivable and modifiable through production of events.

It is important to stress that in our model, norms, agreements and sanctions are expressed as complex structures, meaning that they can be deployed as objects in an institutional space. Institutional spaces can be folded inside other spaces or can be distributed across more than one space creating complex topologies. In this paper we show how a space can be created inside another but we do not elaborate the details of possible dependencies between different institutional spaces. We assume that norms of a father space are not propagated to a child space inside it. Each institutional space is discrete and distinct.

In general, in normative systems, agents' interactions can create new institutional realities (e.g. new EIs). In our model, each time a new EI is to be born, a new institutional space is being created, which includes all the norms, the objects and the agents of the institution, which combined together constitute a first-class representation of an EI.

3.2 Modeling First-Class Electronic Institutions with the Object Event Calculus and C-Logic

In order to describe the dynamics of our meta-model we use the Object Event Calculus (OEC) formalization. The Object Event Calculus is a dialect of the Event Calculus (EC) [18] that is suitable to represent the evolution in time of complex structures by means of events. The main advantage of OEC is that it determines the state of an object by assigning values to its attributes. Based on this property, it deals with the evolution of an object over time, parameterizing its attributes with times at which these attributes hold various values.

The Object Event Calculus predicates we use for the purposes of this paper are shown below:

(C1) holds_at(Id, Class, Attr, Val, T)← happens(E, Ti), Ti ≤ T, initiates(E, Id, Class, Attr, Val),
not broken(Id, Class, Attr, Val, Ti, T).
(C2) broken(Id, Class, Attr, Val, Ti, Tn)← happens(E, Tj), Ti < Tj ≤Tn, terminates(E, Id, Class, Attr, Val).
(C3) holds_at(Id, Class, Attr, Val, T)← method(Class, Id, Attr, Val, Body), solve_at(Body, T).
(C4) attribute_of(Class, X, Type)← attribute(Class, X, Type).
(C5) attribute_of(Sub, X, Type)← is_a(Sub, Class), attribute_of(Class, X, Type).
(C6) instance_of(Id, Class, T)← happens(E, Ti), Ti ≤ T, assigns(E, Id, Class), not removed(Id, Class, Ti, T).
(C7) removed(Id, Class, Ti, Tn)← happens(E, Tj), Ti < Tj ≤ Tn, destroys(E, Id).
(C8) assigns(E, Id, Class)← is_a(Sub, Class), assigns(E, Id, Sub).
(C9) terminates(E, Id, Class, Attr, _)← attribute_of(Class, Attr, single), initiates(E, Id, Class, Attr, _).
(C10) terminates(E, Id, _, Attr, _)← destroys(E, Id).
(C11) terminates(E, Id, _, Attr, IdVal)← destroys(E, IdVal).

Clauses C1-C2 provide the basic formulation of OEC deriving how the value of an attribute for a complex term holds at a specific time. Clause C3 describes how to represent derived attributes of objects treated as method calls computed by means of a solve_at/2 meta-interpreter as specified in [17]. C4-C5 support a monotonic inheritance of attributes names for a class limited to the subset relation. As C1-C2 describe what holds at a specific time, C6-C7 determine how to derive the instance of a class at a specific time. The effects of an event on a class is given by assignment assertions; the clause C8 states how any new instance of a class becomes a new instance of the super-classes. Finally, deletion of objects is catered for by clauses C9-C11. C9 deletes single valued attributes that have been updated,while C10-C11 delete objects and dangling references.

All the structural entities of our meta-model are considered OEC objects and the relationships between them are depicted in Fig. 1

To represent the state of the entities at a given time, we will use the C-logic formalism [10] as it is a convenient formalism to represent complex structures and it has

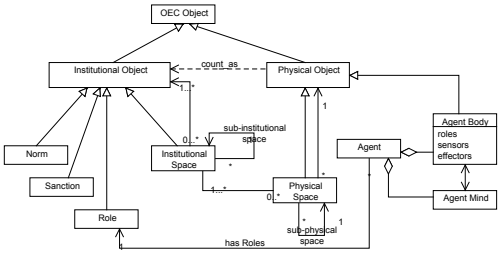


Fig. 1. Entity Categories

a direct translation to the OEC. Complex object descriptions are considered as collections of atomic properties. An object with several attribute labels can be considered as a collection of several atomic formulas. According to the definition of spaces we have introduced, to describe the state of an institutional space at a given time we utilize the following C-logic structure:

institutional_space:is1[
agents ⇒ { agent:a1[roles ⇒ {role:r1, role:r2}], agent:a2[roles ⇒ {role:r1, role:r3}]},
institutional_objects ⇒ {norm_object:11, inst_object:o2, inst_object:o3},
institutional_spaces ⇒ { institutional_space:s2, institutional_space:s3,institutional_space:s4}]

that means that is1 is an institutional_space, which has a set of agents a1, a2, a set of institutional objects that the agents can manipulate in the EI and a set of sub institutional spaces. We can translate the C-logic term above to the following first order logic clauses that we can query utilizing the predicates of the OEC:

is_a(is1, institutional_space). attribute(institutional_space, agents, multi).
attribute(institutional_space, institutional_objects, multi). attribute(institutional_space, institutional_spaces, multi).
time(e1,1). instance(is1,institutional_space, start(e1)). object(is1,agents, a1,start(e1)).
objects(is1,agents,a2,start(e1)). object(is1,institutional_objects, o1,start(e1)).
objects(is1,institutional_objects,o2,start(e1)). object(is1,institutional_spaces, s3,start(e1)).
objects(is1,institutional_objects,s4,start(e1)).

Similarly, the following C-logic structures:

power:p1[mediates ⇒ open_auction:Ev[actor ⇒ IDActor]@T, check_role ⇒ {IDActor, employee}]
sanction:s1[agent ⇒ ag1, credits ⇒ 200]

describe respectively a power rule p1, that mediates events of class open_auction, by checking the power of an agent IDActor that enters the EI as an employee to open an auction at time T, and a sanction s1 of 200 credits, applied to agent ag1. We will show later in this paper how such norms and sanctions are applied when the agents execute an action.

3.3 Evolution of Institutional Spaces

To represent our EIs as first-class abstractions, we will need to define how to represent the state of an EI, how to perceive its state and the state of the agents taking part in

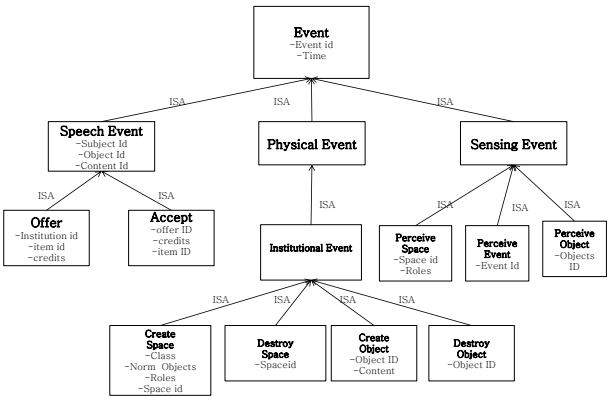


Fig. 2. Events Hierarchy

the interaction, and how to represent the events. In this Section we will illustrate our approach in defining EIs by means of the OEC.

The event schema that we take into consideration in our system is shown in Fig.2. We distinguish between three kinds of events that are speech events, physical events and sensing events. This distinction is not new and it was already presented in [7], in this paper we further extend the hierarchy of events introducing *institutional events*. Institutional events are considered physical events. This does not go against Searle's definition of social events [23], as, despite the fact that the institutional events modify institutional entitites, they actually change the state of the agent environment acting as regular physical events. Institutional events include the creation/deletion of institutional spaces and objects and are necessary for the construction of every new first-class electronic institution that happens during the evolution of the MAS. Event descriptions are specified as complex terms and are perceivable by any agent inside the institutional space. This property of the events allows the agents to understand every action that happens inside their institutional context. For example, the event description below:

open_auction:e14[actor ⇒ ag1, auction ⇒ auction:au1[item⇒ medical_item:item1]].

represents an institutional action of agent ag1 who attempts to open an auction about an item item1 of class medical_item. We will see later, how such an action is executed by the agent that causes the event to happen. For the time being, we will assume that the event has happened and we will show how the entities' state in the agent environment will evolve as a result of the happening of this event. To do this we need to define domain specific initiates and terminates clauses, as shown below:

assigns(E,Obj, auction) ← open_auction:E, auction_of(E,Obj).
initiates(E, Au, auction, item, I) ← open_auction:E [item ⇒ I].

in this way the assigns/3 domain dependent clause above deals with the creation of an auction while the initiates/5 clause assigns an attribute item to the newly created institution. The specification would need also the definition of destroys/3 and terminates/5 clauses to deal with the destruction of an object and termination of an attribute; this is handled in the OEC by the clause C9 shown in Section 3.

3.4 Acting and Perceiving Inside Institutions

The representation in terms of C-logic structures of the EIs allows us to have multiple institutions recursively embedded within each others. In order to act within an institutional space, the agents have to be aware of the space where they want to perform an action. For the purposes of this paper, we do not consider dependencies between institutional spaces.

Moreover, the agents' actions are going to be mediated by the regulative rules of the institution as we have already mentioned. As a consequence we say that in order to be performed within an EI, an action has to be attempted in that EI first. We specify how the EI evolves in time by means of assertion of events, where we keep the events description separated from the attempt.

attempt(e14, 120).
do:e14[actor ⇒ ag1, act ⇒ open_auction:m1[institutional_space⇒ IS1]].

In particular, through the rule H1a below we say that in order to happen within the EI the event has to be attempted, the agent producing the event has to have the power to produce the event and the event must be permitted.

H1a)happens(Event, T)← attempt(Event[institutional_space ⇒ IS], T), power(Event,T), permitted(Event, T).
H1b)happens(Event, T)← happens(Event*,T), counts_as(Event*[institutional_space ⇒ IS],Event,T).
H1c)happens(sanction:Event, T)← obligation(Event* @T*,T), T* =T.

As a consequence of defining the happens/2 relation in this way, agents must be aware of the normative systems where they produce events. The rule H1b handles those cases when an event produced outside the normative system, like a physical event in the agent environment, has an effect on a normative system. To achieve this we make use of the counts_as/3 predicate, which states that if an event Event* happens at time T, then also another event Event in relation to an institution identified by IS happens too. The rule H1c specifies that if an obligation has not been satisfied until T, where @T means "at time T", a sanction event happens. We specify further the predicates to enforce the norms of the institution as follows:

H2)obligation(Ev[institutional_space ⇒ IS],T)← instance_of(IS,institutional_space,T), holds_at(IS,norm_object,Oid,T),
instance_of(Oid,obligation,T), apply_norm(Oid,Ev,T).
H3)permission(Ev[institutional_space ⇒ IS],T)← instance_of(IS, space,T), holds_at(IS,institutional_space,norm_object,Oid,T),
instance_of(Oid,permission,T), apply_norm(Oid,Ev,T).
H4)power(Ev[institutional_space ⇒ IS],T)← instance_of(Sid, institutional_space,T),
holds_at(Sid,institutional_space, norm_object, Oid,T), instance_of(Oid,power,T), apply_norm(Oid,Ev,T).

The clauses H2), H3), H4) specify the concepts of power, permission and obligation, that define three distinct kind of norms. The predicate apply_norm/3 is a meta-interpreter that takes the norms in form of objects and check them against the events produced. To express how perception takes place in the EIs, we define the H5) and H6) clauses.

H5)notify(Class:E, Sensor, T) ← happens(E, T), E[institutional_space ⇒ IS], holds_at(IS,agent,Ag,T),
holds_at(IS,owns,Sensor,T), holds_at(Sensor,senses,Class,T).
H6)perceiveE, S, T) ← happens(E, T), perceive_institutional_space(E), E[sensor_of⇒ S, focus⇒ Focus,
institutional_space ⇒ IS].

H5) specifies that whenever an event happens within an institutional space, such event is notified to the agents that are part of such space if they have a sensor that is capable to perceive such events. H6) specifies how an agent can focus on a particular institutional space and perceive its properties, where the solve_at/3 predicate returns a variable substitution of the variables in Focus, if any. The implications of rule H6) is that the agents deployed in the agent environment and taking place in an institutional space can perceive the institutional entities, such as agreements, sanctions and norms, in the institutional space.

4 Applying MANET to an e-Health Marketplace

4.1 What Is the e-Health Marketplace

During the last decade there have been many efforts towards the reduction of health costs [24] [28]. The public health care system rapidly is absorbing an ever-increasing share of the gross domestic product [22]. According to the latest available data, hospital costs account for approximately 35% physicians, 25% drugs, 15% medical equipment and supporting IT tools while the rest 25% concerns various secondary health costs. Although there have been numerous research projects trying to reduce the hospital costs without losing the quality of offered services, the cost of drug and medical equipment supplies is continuing to increase.

An automatic negotiation mechanism would enhance the cooperation of medical units as well as the optimal use of their medical supplies, leading to a reduction of medical supplies costs. Such negotiation platform for hospitals and medical units could benefit if designed as an open multi-agent system.

To illustrate our meta-model, we take as a motivating case study a network of hospitals which need to trade about several different medical items. The general medical challenges that we want to address with our negotiation system are the following ones:

- How do we exploit blood and medicament overplus?
- How do we exploit medicaments whose expiration date is approaching? How is it possible to supply them at another hospital unit?
- How do we enhance the cooperation and the coordination of hospital units during urgent incidents?
- How do we reduce the costs for drug supplies at the hospitals?
- How do we ensure the cost and time efficient accomplishment of inter-hospital requests?

In particular, our concrete objective is to create a negotiation platform that supports the trading of three different categories of products such as (a) medicaments, (b) blood, (c) medical equipment. This negotiation platform is considered as a marketplace where multiple Dutch auctions can occur simultaneously. Each hospital can start an auction in order to trade a product or it can join an already running auction in order to express interest for buying a product. A number of legal restrictions can apply at a marketplace between different hospital units. Some of them are related to the rights of hospitals to re-sell medical supplies. How we could overcome these legal obstacles is still a point for further study. For the purpose of this paper we propose a solution based on the replacement of real monetary units with virtual ones. Each product belonging to one of the three categories mentioned above, will be negotiated with a starting price expressed in terms of credits. This implies that each hospital unit will have an account balance with the other hospitals on credits. The introduction of the notion of credits instead of real monetary units is crucial in order to avoid legal restrictions on trading of medical supplies.

Given this setting, we do not make any assumption on the kind of entities performing the negotiation: they can be either human entities or software agents interacting with the negotiation platform, although it is important to say that in an emergency scenario one would expect that the trade is handled by human agents rather than from software agents.

The case of e-Health marketplace is a typical example of an open system whom organization can be modeled by EIs and thus it is suitable in order to depict the functionalities of the MANET framework.

4.2 Formalizing the e-Health Marketplace

In order to present the properties of the MANET meta-model we model here a scenario of the e-Health Market place, making use of the general purpose rules presented in Section 3 and we add a set of domain dependent axioms to deal with the evolution of an EI representing a Dutch auction in order to sell medicaments in am e-Health

marketplace. In particular, we adapted the Dutch auction as presented in [13] to our formalism based on the OEC and we introduce norms expressed in terms of objects of the Object Event Calculus formalism. Fig. 3 shows the life-cycle of a Dutch auction within the auction house agent environment.

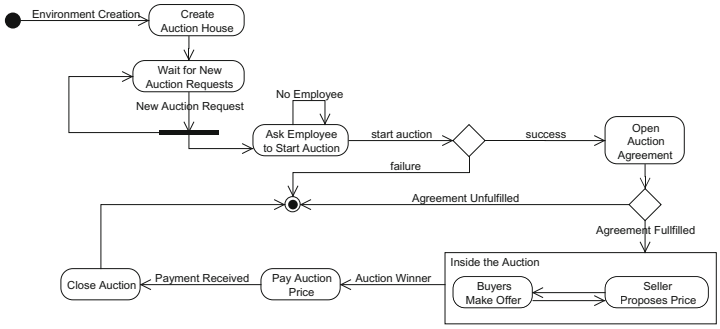


Fig. 3. Auction House Environment State Chart

The Dutch auction electronic institution defines a set of roles for the performance of institutional actions. The roles define the powers that the agents have in the institution. These roles, for the purposes of the e-Health marketplace example, are: a) *Employee*: it is an agent representing the auction house agent environment and that is entitled to open auctions. The agent having this role can also run an auction assuming the role of auctioneer for that auction; b) *Participant*: it is an agent that can express interest for an auction, becoming buyers within the auction; c) *Buyers*: it is an agent that is participating in an auction in the auction house agent environment trying to buy an item of interest; d) *Auctioneer*: it is an agent that coordinates an auction on behalf of a seller agent; e) *Seller*: it is an agent that delegates an auctioneer to sell an item in the Dutch auction.

In Fig. 4, once the auction house is created, the environment waits for the opening of an auction. Once a seller contacts an employee agent to open an auction, the employee agent creates an agreement institutional object in the agent environment. This agreement is perceivable by all the agents inside the auction house, which can modify its attributes only with institutional actions. In C-logic terms this agreement object is described as follows:

```
agreement:c1[
        object ⇒ medic_item:b1, debtor:a1[ roles ⇒ {role:employee, role:seller}],
        creditor:a3[ roles ⇒ {role:seller, role:auctioneer}], minimum_price ⇒ 200, deadline ⇒ 2000,
        participants ⇒ {agent:aid1, agent:aid2 ... agent:aidn}]
```

The term above specifies that an agent a1 is going to open an auction before time 2000 for a medical item b1. When an agreement is created, it can be observed from the agents populating the environment, which can express their interest in participating in the auction by modifying the participants attribute of the agreement object with an institutional action. In particular, when the deadline of the agreement expires, we utilize a count_as/3 as follows:

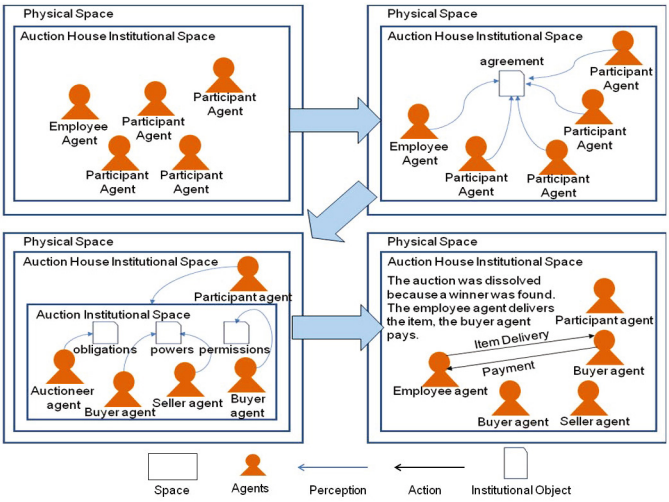


Fig. 4. Interaction in the Auction House Agent Environment

count_as(participate:Ev, assign_role:Ev*, T)← actor_of(Ev,AID),instance_of(C,agreement,T),
instance_of(C,institution,IS,T),holds_at(C,participant, AID,T), role_of(Ev*,buyer), institution_of(Ev*, IS).

to specify that the participate event Ev counts as an assign role event Ev* in the newly created EI for the auction.

The powers of the agents are constrained by the permission norms in the EI: for example an auctioneer is authorized to open an auction only if its starting time has elapsed and if there are at least two agents registered as participants. Fig. 4 represents the interaction taking place in the agent environment represented by the auction house where the auctions are created and dissolved. In particular, as defined in the auction life-cycle in Fig. 3, the e-Health auction is dissolved when an agent wins the auction offering a price that matches the current offer of the seller. To handle the evolution of the auction within the agent environment represented by the auction house, we utilize the following norms:

N1)power:n1[mediates ⇒ start_action:Ev[auctioneer⇒ agent:A, item ⇒ O, starting_price ⇒ P,
institutional_space ⇒ IS]@T, check_role ⇒ {A, employee, T}]
N2)power:n2[mediates⇒change_price:Ev[auctioneer⇒agent:A, item ⇒ O, new_price ⇒ Price,institutional_space ⇒ IS]@T,
check_role ⇒ {IS, A,auctioneer, T}]
N3)permission:n3[mediates⇒ change_price:Ev[auctioneer⇒agent:A,item ⇒O,new_price ⇒ Price]@T,hasItem⇒{IS,O T}
check_role ⇒ {IS,A,auctioneer, T}, hasPrice ⇒ {IS, O, CurrentPrice, T}, lessThan ⇒ {Price, CurrentPrice}]
N4)obligation:n4[mediates ⇒ assign_item:Ev[auctioneer ⇒ Auc, item ⇒ O, buyer ⇒ Buyer, institutional_space ⇒ IS]@T,
lastOffer ⇒ {IS,Buyer,O, LastOffer}, currentPrice ⇒ {IS, O,CurrentPrice}, equal ⇒ { LastOffer,CurrentPrice }]
N5)obligation:n5[mediates ⇒ pay:Ev[buyer ⇒ Buyer, amount ⇒ LastOffer, item ⇒ O, institutional_space ⇒ IS]@T,
isAssigned ⇒ { IS, Buyer,O,T }, currentPrice ⇒ { IS,O,Price,T }, equal ⇒ LastOffer,Price]

Norm N1 specifies that an agent has the power to start an auction in the auction house space when it is an employee for the auction house, while norm N2 and norm N3 express the power of an agent to change the price of an item within an auction space in which the agent is taking part with the role of auctioneer, and the permission to change the price from the point of view of the auction if the auction has that item and the new price is less than the previous one. Norm N4 expresses the obligation of the auctioneer to assign an item to the winner of the auction, while norm N5 expresses the obligation of

a buyer agent to pay for the item assigned by the auctioneer. In the case that the events produced by the agents respect the norms of the institutional space, then the evolution of the Dutch auction institutional space is handled in terms of initiates/4 and terminates/4 clauses that modify the attributes of the Dutch auction whenever an event takes place. For example, the following clauses:

initiates(change_price:Ev, AuID, lastoffer, NewOff)← time(Ev,T), offer(Ev, NewOff), auction(Ev,AuID), value(NewOff, NVal), holds_at(AuID, lastoffer, OldOff, T), value(OldOff, OVal), NVal < OVal.

terminates(makeoffer:Ev, AuID, lastoffer, _)← initiates(makeoffer:Ev, AuID, lastoffer, NewOff).

state that a new offer, is considered the last offer, only if the value of the offer is less than the previous offer. Finally, we introduce a domain dependent count_as/3 to deal with the case of a buyer agent leaving the institutional space of an auction before having paid:

count_as(leave_auction:Ev[institutional_space ⇒ IS],sanction:Ev*, T)← actor_of(Ev,AID), obligation(pay:Ev**[institutional_space ⇒ IS],T), actor_of(Ev**,AID), institution_of(Ev*, IS), credit_of(Ev*,200).

The count_as/3 above specifies that an agent leaving while an obligation of paying holds in the EI will be sanctioned of 200 credits. A further count_as/3 clause has been defined to handle the exception of an auctioneer not delivering the good after the auction, but we omit it as it is similar to the clause above.

5 Implementation Issues

For the implementation of the normative systems we adopted a logic programming approach due to the formal and declarative semantics of our model and we implemented it as a Prolog theory. In particular, we utilized a version of the OEC described in [17], which is based on caching the periods of time in which an attribute of an object holds. For example the top-level implementation of the holds_at/4 predicate is specified as follows using object/4 facts:

holds_at(ID,Attr,Val,Time):- object(ID,Attr,Val,start(Ev_1)), time(Ev_1,T_1), T_1 < Time, not (object(ID,Attr,Val,end(Ev_2)), time(Ev_2,T_2), T_2 ≤Time, T_2 > T_1.

where the object/4 assertions store when an attribute has been initiated/terminated at a certain time.

A similar implementation was used for the instance_of/3 predicate, using instance/3 facts in the Prolog engine. This representation brings the advantage that indexing can be performed on both the time interval and the object identifier, meaning that the time to retrieve the attribute of an object is O(1), once the identifier and the interval are known as in our specification. Using this approach, Prolog is speeding up the computation of OEC predicates [27].

When an agent wants to produce events in MANET, it can call the act/2 predicate, that is specified as follows:

act(E,T):- power(E,T), permission(E,T), produce(E,T).

where power/2 and permission/2 check the event against the existing powers and permission within the institution where it has been produced, while the produce/2 predicate modifies the state of the Prolog database according to the event.

An example of the state of a norm of the normative system to create an auction in the auction house agent environment can be expressed as follows:

```
instance(r1:[IDS,ID,auctioneer, T], power, start(e1)). object(r1:[IDS,ID,auctioneer, T], mediates, open_auction, start(e1)).
object(r1:[IDS,ID,auctioneer, T],template, do(ID,open_auction, [property(institution, IDS)]):T,start(e1)).
object(r1:[IDS,ID,auctioneer, T],check_role,[IDS,ID, auctioneer,T],start(e1)). time(e1,1).
```

The state above specifies that there is an instance of a norm that mediates an event of class open_auction, where the template attribute defines the mediated event. The norm also specifies that it calls the check_role/4 predicate to check if the role of the agent performing the event is the one of auctioneer. Notice that we append the variables that will be called in the check_role/4 predicate in the identifier of the rule, so that we can instantiate their value in the apply_norm/2 meta-predicate as specified below:

```
attempt(E,T):- power(E,T), permitted(E,T), add(E,T).
power(E,T):- E = do(Actor,EventClass, Elements), member(property(institution, IDS), Elements),
        instance_of(IDS, institution,T), holds_at(IDS, rules, ID:Vars, T), instance_of(ID:Vars,power,T),
        holds_at(ID:Vars,template, E:T,T), not(not(apply_norm(ID:Vars,T))).
apply_norm(ID:Vars,T):- holds_at(ID:Vars, Attr, Vars, T), append([Attr],Vars, Var2), Pred =.. Var2, Pred.
```

The attempt/2 predicate implemented above checks if the agent has the power and the permission to produce an event in the environment. If this is the case the add/2 predicates add object/4 or instance/3 assertions to the Prolog database, according to the effects of the event. The power/2 predicate, checks if there is a norm that specifies if the agent has the power to produce a certain event with respect to a certain institution. To do so, the power/2 predicate utilizes the apply_norm/2 meta-predicate to check if the norm specifies any constraint that prevents the agent from performing the action. For example, we implement the check_role/4 constraint as follows:

```
check_role(IDS, ID,Role,T):- instance_of(ID,agent,T), instance_of(IDS,institution,T),
                holds_at(IDS, roles,RID,T), instance_of(RID,Role,T), holds_at(RID, agent, ID,T).
```

The check_role/4 predicate implemented above checks if an agent identified with the variable ID, has a certain role Role in an EI IDS at a certain time T.

Finally, we create spaces containing norm objects as following:

```
produce(do(AID,open_auction,[property(institution, IDS), property(medicament, StartingPrice), AgentList]), T):-
    gensym(ev,Ev),gensym(auction, AucHID), gensym(permission, Perm),
    assert(instance(AucHID, institutional_space, start(Ev))),
    assert(instance(AucHID, auction, start(Ev))), assert(object(AucHID, employee,AID, start(Ev))),
    assert(object(IDS, auction,AucHID, start(Ev))), assert(object(AucHID, state,open, start(Ev))),
    assert_agent_roles(AucHID,Ev,AgentList),
    assert(instance(Perm:[AucHID,IDS,AID,IDBuyer,Time],permission, start(Ev))),
    assert(object(Perm:[AucHID,IDS,AID,IDBuyer,Time], mediates,do(AID,declare_winner,
            [property(auction, AucHID), property(auction_house, IDS), buyer(IDBuyer), sellingPrice(Price)]):Time, start(Ev))),
    assert(object(AucHID, permission, Perm:[AucHID,IDS,AID,IDBuyer,Time], start(Ev))),
    assert(time(Ev,T)).
```

In the predicate above, an object of type permission is created as an attibute of the auction that is being instantiated after the production of the event open_auction.
Within the permission/2 and power/2 predicates we check about the norm objects of an institution by using the apply_norm/2 predicate:

```
apply_norm(ID:Vars,T) :- holds_at(ID:Vars, predicate:Attr, Vars, T), append([Attr], Vars, ListForm),Pred =.. ListForm ,Pred.
```

Such a meta-predicate makes use of the =.. Prolog operator to build a query for the Prolog database using the norm objects. This allows to create dynamic auction spaces where the norms are applied only when the object exists and not to every single event, meaning that we can decentralise the check on norms to the single institutional spaces, as if the institutional spaces represented the boundaries for the production of the events.

6 Related Work

Fornara et al. have developed OCeAN [14], a meta-model for the specification of electronic institutions, and an Agent Communication Language (ACL) to model open interaction systems where heterogeneous software and human agents interact. The OCeAN meta-model consists of the following components: (i) constructs to define the core ontology of an institution, (ii) roles and of events; (iii) the counts-as relation, which is necessary for the concrete performance of institutional actions; (iv) and norms. One difference between MANET and the OCeAN meta-model is that we consider institutions as first-class entities, which allow the perception of their components (e.g. norms, objects and sanctions) that are also described as first-class entities. Another difference is related to the types of events that are possible inside an institution. In the OCeAN meta-model only communication events are considered, whereas in our approach we define a more detailed schema of events in order to describe all the possible situations during the evolution of an open MAS.

Another work with a similar approach to ours, is this of Cardoso and Oliveira [9]. The authors consider EIs as a software framework which consists of a set of services and a normative environment. The role of the services is to allow agents to create organizational structures ruled by a set of mutual commitments and norms. In MANET, we try to solve the problem of creating organizational structures with a similar way, by using first-class institutional objects which exist in the agent environment. The affordances of the objects allow the agents to perceive them and use them in order to create new institutional reality. The way the objects can be used is regulated by the norms of the space wherein the obect reside.

Artikis and Sergot in [3] present a model of executable specifications of open MAS where open MAS are considered instances of normative systems. The authors represent the social constraints (laws) of the system in terms of physical capabilities, institutional power, permission and prohibition as well as sanctions and enforcement policies. In our model we adopt a very similar model of institutional rules based on powers which are dependent on permissions, obligations and sanctions. However, in our work we consider institutional rules as first-class entities which can be observed by the agents, allowing them to reason about the normative constraints of the open MAS.

In [27] Urovi and Stathis define the MAGE framework. MAGE uses the OEC formalism to represent games as first-class entities that evolve in time. Such games are interconnected in a hierarchy composed of atomic games and composite games. The state of the composite games is defined by the relationships between the atomic games and their transitions and the agents can perceive the legal actions in a game at a certain time. MANET institutional spaces correspond to MAGE games which also evolve due to the production of events. The main difference between MANET and MAGE is that we include the possibility of defining institutional objects, such as norms and agreements, allowing for norms to have a structure and be perceivable, meaning that the agents can reason about whether or not complying with a norm is to their best interest.

Also related to our work is the work of Piunti et al. to unify in one model the concepts of agents, organizations and environments [21]. This model allows for designing and programming an environment in terms of a dynamic set of first-class computational entities called artifacts, collected in workspaces. Artifacts represent resources and tools

that agents can dynamically instantiate, share and use to support their individual and collective activities. The notions of artifacts and workspaces have similarities with these of objects and spaces. But unlike Piunti et al., spaces in our approach are not just the containers of agents and objects modeling the locality of the application domain but in contrast they are the main enforcers of the law and regulators of the MAS evolution.

Finally, Boissier and Fred [6] proposed a framework for normative organizations where organizational artifacts are used in order to instrument the multi-agent environment and the organizational entities living whithin them. These artifacts can also show as observable properties information related to the current status of the norms given the agents behaviour related to organizations they are linked to. In MANET we follow a different line. The norms are described as first-class objects and as result their status and state are dirrectly obsvervable inside the organization.

7 Conclusion and Future Works

We presented a meta-model that describes institutions as social agent environments [29] by extending upon the OEC formalism [17], based on the concept of agent environment as presented in [7]. In particular we presented a model that can handle the life-cycle of multiple institutions at runtime, where the institutions are represented as first-class objects that the agents can perceive. Moreover, our model supports the definition of institutional objects, such as agreements, sanctions and norms that the agents can perceive as part of an electronic institution. We presented this model utilizing an e-Health marketplace based on Dutch auctions as a motivating example.

There are several directions that are worth exploring for future work. First of all, we plan to extend our framework towards an application independent model describing all the properties of electronic institutions and whose run-time specifications are fully explained.

Secondly, we want to handle dynamic norm change within the institution when an agent society requires it due to an external exception. For this reason we have to provide a complete methodology for the perception of norms by the agents. As recognized by Artikis et al. in [2], learning the rules of an institution is recognized as a problem, as a consequence we plan to investigate machine learning approaches that can make use of our model to create cognitive agents capable of learning how to interact within multiple heterogeneous institutions.

Finally, an important future contribution would be the definition of an approach for managing institutional spaces' interdependencies. In particular, we plan to propose an approach based on the first-class concepts of space and object for tackling two basic EIs interdependency questions: i) how is the behaviour of an agent affected when it is simultaneously participating in more than one institutional space having conflicting norms?; ii) how is it possible to monitor and mediate (if applicable) the behaviour of an agent participating in more than one institutional space?

Acknowledgements. Our work is partially supported by the SER project Open Interaction Frameworks: Towards a Governing Environment under the Agreement Technologies COST Action IC0801.

References

1. Ägotnes, T., van der Hoek, W., Wooldridge, M.: Normative system games. In: AAMAS 2007: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1–8. ACM, New York (2007)
2. Artikis, A., Paliouras, G., Portet, F., Skarlatidis, A.: Logic-based representation, reasoning and machine learning for event recognition. In: DEBS, pp. 282–293 (2010)
3. Artikis, A., Sergot, M.: Executable Specification of Open Multi-Agent Systems. Logic Journal of the IGPL 18(1), 31–65 (2010)
4. Barber, K.S., Kim, J.: Soft Security: Isolating Unreliable Agents from Society. In: Falcone, R., Barber, S.K., Korba, L., Singh, M.P. (eds.) AAMAS 2002 Ws Trust, Reputation... LNCS (LNAI), vol. 2631, pp. 224–233. Springer, Heidelberg (2003)
5. Boella, G., Pigozzi, G., van der Torre, L.: Normative systems in computer science - ten guidelines for normative multiagent systems. In: Boella, G., Noriega, P., Pigozzi, G., Verhagen, H. (eds.) Normative Multi-Agent Systems, Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 09121 (2009)
6. Boissier, O., Hübner, J.F.: A framework for normative multiagent organisations. In: Boella, G., Noriega, P., Pigozzi, G., Verhagen, H. (eds.) Normative Multi-Agent Systems, Dagstuhl. Dagstuhl Seminar Proceedings, vol. 09121. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009)
7. Bromuri, S., Stathis, K.: Distributed Agent Environments in the Ambient Event Calculus. In: DEBS 2009: Proceedings of the Third International Conference on Distributed Event-Based Systems. ACM, New York (2009)
8. Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J.A., Esteva, M.: Formalising Situatedness and Adaptation in Electronic Institutions. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS, vol. 5428, pp. 126–139. Springer, Heidelberg (2009)
9. Cardoso, H.L., Oliveira, E.: A Context-Based Institutional Normative Environment. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN@AAMAS 2008. LNCS, vol. 5428, pp. 140–155. Springer, Heidelberg (2009)
10. Chen, W., Warren, D.S.: C-logic of Complex Objects. In: PODS 1989: Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 369–378. ACM Press, New York (1989)
11. Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., Arcos, J.-L.: On the Formal Specification of Electronic Institutions. In: Sierra, C., Dignum, F.P.M. (eds.) Agent Mediated Elec. Commerce. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
12. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: Ameli: An agent-based middleware for electronic institutions, vol. I, pp. 236–243. ACM (2004)
13. Fornara, N., Colombetti, M.: Specifying artificial institutions in the event calculus. In: Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models, pp. 335–366 (2009)
14. Fornara, N., Viganò, F., Verdicchio, M., Colombetti, M.: Artificial institutions: a model of institutional reality for open multiagent systems. Artif. Intell. Law 16(1), 89–105 (2008)
15. Gruber, T.: Collective knowledge systems: Where the Social Web meets the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web 6(1), 4–13 (2007)
16. Hendler, J., Berners-Lee, T.: From the semantic web to social machines: A research challenge for ai on the world wide web. Artif. Intell. 174(2), 156–161 (2010)
17. Kesim, F.N., Sergot, M.: A Logic Programming Framework for Modeling Temporal Objects. IEEE Transactions on Knowledge and Data Engineering 8(5), 724–741 (1996)
18. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gen. Comput. 4(1), 67–95 (1986)

19. Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. In: AAMAS 2009: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 153–160 (2009)
20. Pitt, J., Mamdani, A., Charlton, P.: The open agent society and its enemies: a position statement and research programme. Telematics and Informatics 18(1), 67–87 (2001)
21. Piunti, M., Boissier, O., Hubner, J.F., Ricci, A.: Embodied organizations: a unifying perspective in programming agents, organizations and environments. In: 11th Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems (2010)
22. Reinhardt, U.E., Hussey, P.S., Anderson, G.F.: U.S. health care spending in an international context. Health Affairs 23(3) (2004)
23. Searle, J.R.: The construction of social reality. Free Press, New York (1995)
24. Stanton, M.W.: Reducing costs in the health care system; learning from what has been done. Healthcare Research and Quality (2003)
25. Strachey, C.: Fundamental concepts in programming languages. Higher Order Symbol. Comput. 13(1-2), 11–49 (2000)
26. Stratulat, T., Ferber, J., Tranier, J.: Masq: towards an integral approach to interaction. In: AAMAS (2), pp. 813–820 (2009)
27. Urovi, V., Stathis, K.: Playing with Agent Coordination Patterns in MAGE. In: Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., da Silva, V.T., Matson, E., Polleres, A. (eds.) COIN 2009. LNCS, vol. 6069, pp. 86–101. Springer, Heidelberg (2010)
28. Wasserfallen, J., von Auw, Y.: Cost reduction project in a swiss public hospital. In: Annual Meeting of the International Society of Technology Assessment in Health Care (1998)
29. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1), 5–30 (2007)
30. Wooldridge, M.: Multi Agent Systems. John Wiley and Sons (2002)

Towards Practical Normative Agents: A Framework and an Implementation for Norm-Aware Planning

Sofia Panagiotidi and Javier Vázquez-Salceda

Knowledge Engineering and Machine Learning Group
Universitat Politècnica de Catalunya - BarcelonaTECH
Campus Nord UPC, Edifici K2M
C/ Jordi Girona 1-3, 08034 Barcelona, Spain
{panagiotidi,jvazquez}@lsi.upc.edu

Abstract. Nowadays there is an important increase in the adoption and use of distributed computational solutions which are growing both in size (from tens to hundreds or even thousands of components, computational entities or actors) and in complexity (from closed, static, pre-defined interactions to more open, dynamic ones stablished at run-time). In this scenario one way to tame such complexity is to add a social layer on top regulating or shaping the behaviour of the actors in the system. One of those social abstractions that has been explored in literature is the use of computational models of (social or organisational) norms. Most of these approaches see norms as a way to specify acceptable agent behaviour in some (distributed) context. In literature there is a lot of work on norm theories, models and specifications on how agents might take norms into account when reasoning but few practical implementations. In this paper we present a first step into the implementation of practical normative agents by describing a framework and an implementation of norm-oriented planning. In this framework norms can be either obligations or prohibitions which can be violated, and are accompanied by repair norms in case they are breached. Unlike most frameworks, our approach takes into consideration the operationalisation of norms during the plan generation phase. Norm operational semantics is expressed as an extension/on top of STRIPS semantics, acting as a form of temporal restrictions over the trajectories (plans) computed by the planner. In combination with the agent's utility functions over the actions, the norm-aware planner computes the most profitable trajectory concluding to a state of the world where no pending obligations exist and any (obligation/prohibition) violation has been handled. An implementation of the framework in PDDL is described.

1 Introduction

In the last 20 years software systems have moved from isolated computational nodes with limited networked interactions into distributed solutions across (multiple) networks, computational systems have become more and more complex in nature, resulting in highly complicated interconnected systems. As such systems grow in size to include hundreds (and even thousands of computational components/entities), system designers and managers face the problem on how to tackle the increased complexity and dynamicity of such systems, in special to ensure that the system as a whole an/or its components

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 93–109, 2012.

behave within some acceptable bounds. One approach coming from multi-agent systems research is to add some social models where actors keep some autonomy while their behaviour is somehow regulated in order to produce desirable and avoid undesirable situations. In such models, goals and interactions are usually not specified in terms of the mental states of individual agents, but in terms of social or organisational concepts such as roles (or function, or position), groups (or communities), norms (or regulations or policies) and communication protocols (including ontologies). In these cases, agents are seen as actors that perform the role(s) described by the social/organisational design.

Often the very notion of agent autonomy refers to the capability of an agent to act independently, exhibiting control over its own internal state meaning that an agent needs to anticipate, plan and adopt actions that are in accordance to organisational specifications while, at the same time, optimising its individual outcome. Despite the fact that such norm autonomy is important in complex systems where dynamic decision making is a key element and where conflicts frequently occur, few systems achieve capturing an efficient normative reasoning procedure. In literature there is a lot of work on norm theories, models and specifications on how agents might take norms into account when reasoning. However:

- most of these works focus in the *deliberation step* of an agent's reasoning cycle (that is, deciding WHAT to do from the agents' beliefs, desires and intentions);
- few practical implementations exist that cover the full BDI cycle, as many approaches do not include the *means-ends reasoning step* (that is, deciding HOW to achieve WHAT the agent is aiming for).

In fact in many norm-aware agent implementations the plans are pre-computed off-line, and the means-ends reasoning is reduced into a selection of an appropriate plan to reach some goal or state.

In this paper we aim to move towards a fully normative-aware agent by describing a framework to support practical normative reasoning that can be used by agents to produce their plans. Our approach is based on a extension of the well known STRIPS language to include an extra layer of normative representation that, on top of the domain description, adds norms acting as complex restrictions to the planning problem and influencing the planning mechanism in a way that it produces the most beneficial plans. We also show how a standard PDDL planner is able to receive a domain description and additionally the normative specification and then compute executional paths that consider the restrictions imposed by norms, either conforming to or avoiding to take into consideration while considering any possible sanction.

The paper is structured as follows. In Section 2 we describe a simple scenario that we will use through the paper. Then in Section 3 we describe the semantics of the STRIPS formalisation that we use as basis for our normative model, which is presented in Section 4. In Section 5 we explain how the framework has been adapted for its implementation in PDDL. Section 6 shows code examples on how the framework and the scenario have been translated to PDDL 2.1, and which are the results obtained from a SGPlan 6 planner. Section 7 is devoted to discuss some related work. Finally Section 8 closes the paper.

2 Example Scenario

The example follows a simple scenario where an actor is inside a building (which can be on fire) and the goal of the actor is to find his way out of the building. The actions in this domain can be:

- to get out from the door,
- to break a window (under the condition that some window is not already broken)
- to jump out the window (under the condition that some window is broken).

Norms in this scenario include:

- a norm that prohibits the person to break a window while he is inside the building and, in case he does, the person needs to pay a fine.

Actions are accompanied by a utility function which represents the cost for the agent to execute the action, according to the state of the world when executed. In an evacuation scenario, it is assumed to be rather costly to get out through the door while the building is on fire (as it involves too much risk for the agent's health) and thus the individual might choose to break the window (violating the norm), jump and then pay the fine, as the total cost of following this plan would be less than any other option.

3 STRIPS Planning

Our normative planning framework will extend STRIPS with additional normative elements (see Section 5) in order to allow for normative reasoning within the planning process. In this section, we briefly describe the semantics of the STRIPS formalisation [7].

Definition 1. *We define **F** as set of **fluents** $F = \{f_1, f_2, \ldots, f_m\}$ where fluent f_i is an atomic proposition (propositional property).*

*The state of the world is defined in terms of fluents that hold at the particular situation. We define a **state** to be a (possibly empty) subset of F, i.e. a state is represented by the set of fluents that are true in it. The fluents not in the state are assumed false.*

Each combination of fluents forms a different state, and the union of all the states is a set of states as in Definition 2.

Definition 2. *We define **S** to be the **state domain** (set of all states) occurring from F as $S = 2^F$.*

Definition 3. *Having a set of fluents F as in Definition 1, we define **A** to be a set of domain **actions** (actions with with pre and postconditions) $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ where $\alpha_i =< C_{prec_i}, C_{post_i} >$ and C_{prec_i}, C_{post_i} are conjunctions of fluents or negated fluents from F. Since states are represented by sets of conditions, the **transition function** $\longmapsto$ relative to a domain instance $< F, A >$ is a function $\longmapsto$: $S \times A \to S$ where S is the state domain S occurring from F. If σ, $\sigma' \in S$ and $\alpha \in A$ and $\longmapsto (\sigma, \alpha) = \sigma'$*

then we write $\sigma \overset{\alpha}{\longmapsto} \sigma'$. *The transition function can be defined as follows, using the simplifying assumption that actions can always be executed but have no effect if their preconditions are not met:*

- $\longmapsto (\sigma, < C_{prec_i}, C_{post_i} >)$ *is* $\sigma' \cup \{$*all* m *where* m *belongs to* $C_{prec_i}\} \backslash \{$*all fluents* l *where* $\neg l$ *belongs to* $C_{prec_i}\}$ *if* $\{$*every fluent* j *in* C_{post_i} *belongs in* $\sigma\}$ *and* $\{$*every fluent* k *where* $\neg k$ *in* C_{post_i} *does not belong in* $\sigma\}$
- σ *otherwise*

The function $\longmapsto$ *can be extended to sequences of actions by the following recursive equations:*

$\sigma \overset{[]}{\longmapsto} = \sigma$

$\sigma \overset{[\alpha_1,\alpha_2,\ldots,\alpha_n]}{\longmapsto} = \sigma \overset{\alpha_1}{\longmapsto} \overset{[\alpha_2,\ldots,\alpha_n]}{\longmapsto}$

Definition 4. *A* ***goal*** g *is a pair* $< P, R >$, *where* P *and* R *specify which fluents are true and false, respectively, in order for a state to be considered a goal state.*

In order to proceed to the definition of a plan, we assume the existence of an initial state σ_0 where $\sigma_0 \in S$. Then:

Definition 5. *A* ***plan*** *is a sequence of actions such that the state that results from executing the actions in order from the initial state satisfies the goal conditions. Formally,* $[\alpha_1, \alpha_2, \ldots, \alpha_n]$ *is a plan for goal* $g =< P, R >$ *if the state* $\sigma' = \longmapsto (\sigma_0, [\alpha_2, \ldots, \alpha_n]))$ *is such that* $P \subseteq \sigma'$ *and* $R \cap \sigma' = \emptyset$.

The above means that the state reached by the execution of the actions $\alpha_1, \ldots, \alpha_n$ should lead to a state which satisfies the specification of the goal.

4 Normative Model

Using the domain elements of the STRIPS semantics (fluents, actions, states, goals, plans) as basis, we include norms as the additional elements that will lead to the definition of our normative model.

4.1 Norms

In our framework, norms are defined as follows:

Definition 6. *Given a set of fluents* F *as in Definition 1 we define* Nms *to be a set of* ***norms*** $Nms = \{N_1, N_2, \ldots, N_v\}$ *where* $N_i = \{id, type, C_{act}, C_{deact}, C_{maint}, C_{repair}\}$ *and id is an identificator, type is in* $\{obligation, prohibition\}$ *and* $C_{act}, C_{deact}, C_{maint}$, C_{repair} *are conjunctions of fluents or negated fluents from* F.

The $type$ attribute specifies the deontic operator, expressing the deontic "flavour" of the norm, in our case obligation or prohibition, and thereby establishes whether an agent

should attempt to fulfil the norm or on the contrary avoid the prohibition. The activation, maintenance and deactivating conditions are specified as (partial) state descriptors, denoting the conditions that express when the norm gets activated, violated and deactivated. They add operational information to the norm, to simplify the verification and enforcement of the norm. They work as follows[1]:

- The *activation condition* C_{act} specifies when a norm becomes active, i.e the state of affairs in which the norm is triggered (and must henceforth be checked for completion/violation).
- The *deactivating condition* C_{deact} specifies when the norm has been deactivated, i.e. has no longer normative force.
- The *maintenance condition* C_{maint} is needed for checking violations of the norm; it expresses the state of affairs that should hold all the time between the activation and the deactivation of the norm.
- The *violation deactivation condition* C_{repair} is a violation penalty proposition, a state descriptor indicating the compulsory satisfaction/repair of the penalty of the norm in the case of a violation of a norm.

In essence, when a norm has been activated, has not yet expired and the maintenance condition is not fulfilled, a violation of the norm happens. Whenever a norm is violated, the norm continues to be active but in addition, the agent always has to see to it that the penalty is fulfilled. In this paper we adopt this rather simplistic approach to norm violation handling, still this could be further extended by introducing complete repair norms handling the violation of one

Additionally, a norm might have several active instances at the same time [15]. When the terms in the activating condition hold, the variables are instantiated, creating a new norm instance.

It is also important to note that, as discussed in [2], these kind of tuple representations including norm activation, deactivation and maintenance are as expressive as conditional deontic statements with deadlines (such as the one presented in [6]).

4.2 Definition of Normative Model

Once we have defined the norms in our framework, the normative model can be defined as follows.

Definition 7. *Let a* ***normative model*** *be a tuple* $M = (F, A, Nms)$ *where* F *is a set of fluents as defined in Definition 1,* A *is the set of domain actions (labels) as in Definition 3 and* Nms *a set of norms as defined in Definition 6.*

We define F_{ext} as $F_{ext} = F \cup \{act(N), deact(N), maint(N), repair\}$ where $act, deact, maint, repair$ are special additional fluents indicating the activating, deactivating, maintenance and violation deactivation of a norm N in Nms.

[1] The operational semantics follow our previous work on norm lifecycle semantics, found at [15].

The semantic entailment for the four additional propositions $act(N), deact(N)$, $maint(N)$, $repair(N)$ given a norm $N = \{id, type, C_{act}, C_{deact}, C_{maint}, C_{repair}\}$ is:

- $\Big((M,\sigma) \models act(N)\Big) \Leftrightarrow \Big(\forall f \in C_{act} \cdot f \in \sigma \; and \; \forall \neg f \in C_{act} \cdot f \notin \sigma\Big)$
- $\Big((M,\sigma) \models deact(N)\Big) \Leftrightarrow \Big(\forall f \in C_{deact} \cdot f \in \sigma \; and \; \forall \neg f \in C_{deact} \cdot f \notin \sigma\Big)$
- $\Big((M,\sigma) \models maint(N)\Big) \Leftrightarrow \Big(\forall f \in C_{maint} \cdot f \in \sigma \; and \; \forall \neg f \in C_{maint} \cdot f \notin \sigma\Big)$
- $\Big((M,\sigma) \models repair(N)\Big) \Leftrightarrow \Big(\forall f \in C_{repair} \cdot f \in \sigma \; and \; \forall \neg f \in C_{repair} \cdot f \notin \sigma\Big)$

4.3 Path Temporal Projection

In order to be able to know whether a norm is active or violated, it is not enough to be aware of the current state of affairs. An additional knowledge concerning the norm status at previous states is required. For example, in order to derive that a norm gets deactivated one needs to know not only that the deactivation condition holds at a specific state but also that the norm was active previously. Therefore, such properties need to be defined with respect to an entire sequence of states (trajectory path) visited during the execution of a plan, starting from an initial point.

Definition 8. *Given a set of actions A, a plan* $\pi = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ *and an initial state* σ_0, π *generates the trajectory* $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ *iff for every* $1 \leq i \leq n$: $\sigma_i \xmapsto{\alpha_i} \sigma_{i+1}$

Thus, the entailment for the norm lifecycle properties is defined with respect to a trajectory path produced by a plan in Definition 9.

Definition 9. *The interpretation in* M *of the semantic entailment over a trajectory* $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ *of a path* π *at a state* σ *(norm lifecycle) is as follows:*

- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models active(N) \quad iff \quad \Big((M, \sigma_i) \models act(N) \; and \; \neg deact(N)\Big) \; or \; \Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_{i-1}) \models active(N) \; and \; (M, \sigma_i) \models \neg deact(N)\Big)$
- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models violated(N) \quad iff \quad \Big((M, \sigma_i) \models \neg maint(N) \; and \; (M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models active(N)\Big)$
- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models rep\_active(N) \quad iff \quad \Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models violated(N) \; and \; (M, \sigma_i) \models \neg repair(N)\Big) \; or \; \Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_{i-1}) \models rep\_active(N) \; and \; (M, \sigma_i) \models \neg repair(N)\Big)$

4.4 Definition of the Normative Planning Problem

With all the elements introduced in the prevous definitions, we can now define the exact nature of normative planning. In our framework normative planning is formalized as a planning problem in a domain where there are additional norms which acquire committing (obligation) or preventing (prohibition) force through planning paths, and those paths failing to comply with (some of) the norms need to see to it that a repairing state is reached. In short, the planning problem is defined as finding a plan that:

- the final state achieves the goal,
- there are no pending obligations open
- for all possible violations of obligations and prohibitions that might have occurred, the repair state has been accomplished.

Formally, given an initial state σ_0, a goal g and a normative model $M = (F, A, Nms)$ we are looking for a plan $\pi = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ generating the trajectory $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ where:

- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \models g$
- Forall prohibitions $P \in Nms$: $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \not\models rep\_active(P)$ (at final state the repair norm is not active)
- Forall obligations $O \in Nms$: $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \not\models active(O)$(at final state the obligation is not active) and $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \not\models rep\_active(O)$ (at final state the repair norm is not active)

In modern PDDL versions, actions can have an associated cost function which might be combined with conditional effects. This allows for paths (plans) to have a total cost. We introduce action costs in our framework in order to be able to evaluate not only the norm conformance but also to calculate most beneficial paths while taking into account compliance to or violation and reparation of norms. When defining what the norm repair (penalty) state is, we require that any plan-solution that violates a norm also takes the necessary steps to achieve its repair state. By giving the appropriate cost function to the actions, the planner is able to determine the choice between a path that complies to a norm and a path that violates it and afterwards compensates by reaching the repair state. In this way, such complex reasoning over conformance to the soft restraints imposed by norms becomes part of the planning problem rather that an external issue.

5 Implementation in PDDL

We introduce an intermediate state S'_i (which the planner is forced to include between all actions for every produced plan), which comes after state S_i and before state S_{i+1} (see Figure 1). That is so that we are able to implement the update of the norm status (activation, violation and activation of the repair norm), which, as seen in Section 4 is dependent on the previous state of affairs as well as the current one. This technique is commonly applied in PDDL domains so as to be able to express predicates only in terms of the previous state. In this way, the calculation of the norm lifecycle happens in two stages. In the first, the effects of the action are evaluated (state S'_i) while norm' status remains the same as it was before taking into consideration the action's effects. In the second step (state S_{i+1}) the norm's status is evaluated. Figure 1 depicts an example of the intermediate states introduced in the normative planner.

Below we detail the derivation rules for the norm lifecycle and discuss them.

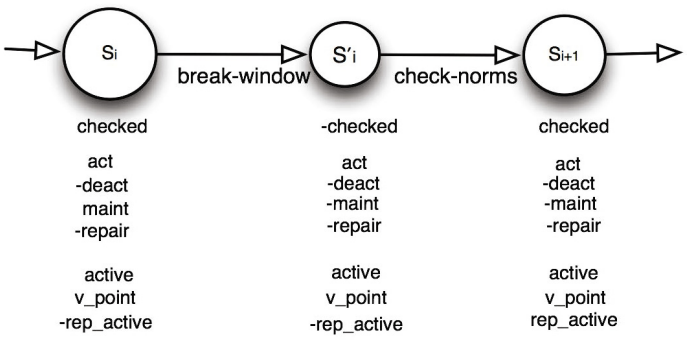


Fig. 1. Intermediate States

(1) $\mathtt{active}(S_{i+1}) = \mathtt{active}(S'_i) \wedge \neg\mathtt{deact}(S_{i+1}) = \mathtt{active}(S'_i) \wedge \neg\mathtt{deact}(S'_i)$[2]

(2) $\mathtt{active}(S_{i+1}) = \mathtt{act}(S_{i+1}) \wedge \neg\mathtt{deact}(S_{i+1}) = \mathtt{act}(S'_i) \wedge \neg\mathtt{deact}(S'_i)$

(3) $\neg\mathtt{active}(S_{i+1}) = \mathtt{deact}(S_{i+1}) = \mathtt{deact}(S'_i)$

(1), (2) and (3) reflect definition 9. They state that a norm is active either if it was active in the previous states and the deactivation condition cannot be inferred, or, if the activation condition can be inferred at the current state and the deactivation condition cannot be inferred. In any other case, the norm is not active.

(4) $\mathtt{v\_point}(S_{i+1}) = \neg\mathtt{v\_point}(S'_i) \wedge \mathtt{active}(S_{i+1}) \wedge \neg\mathtt{maint}(S_{i+1})$

(5) $\neg\mathtt{v\_point}(S_{i+1}) = \mathtt{v\_point}(S'_i)$

(4) and (5) introduce an implementation detail, the `v_point`, which indicates the state where a violation first occurs. This occurs when a norm is active, the previous state was not a violation state and the maintenance condition cannot be inferred at the current state.

(6) $\mathtt{rep\_active}(S_{i+1}) = \mathtt{v\_point}(S_{i+1}) \wedge \neg\mathtt{repair}(S_{i+1})$

(7) $\mathtt{rep\_active}(S_{i+1}) = \mathtt{rep\_active}(S'_i) \wedge \neg\mathtt{repair}(S_{i+1}) = \mathtt{rep\_active}(S'_i) \wedge \neg\mathtt{repair}(S'_i)$

(8) $\neg\mathtt{rep\_active}(S_{i+1}) = \mathtt{repair}(S'_i)$

(6), (7) and (8) reflect definition 9. They state that the reparation of norm is active either if it was active in the previous states and the repair condition cannot be inferred, or, if there has been a violation state and the repair condition cannot be inferred. In any other case, the norm reparation is not active.

[2] Here we only use the state as parameter for the predicate "active" and deliberately omit the norm (that we used in the definitions in Section 7) for reasons of simplicity.

(4)+(1) -->

(9) $\texttt{v\_point}(S_{i+1}) = \neg\texttt{v\_point}(S'_i) \wedge \texttt{active}(S'_i) \wedge \neg\texttt{deact}(S'_i) \wedge \neg\texttt{maint}(S'_i)$

(4)+(2) -->

(10) $\texttt{v\_point}(S_{i+1}) = \neg\texttt{v\_point}(S'_i) \wedge \texttt{act}(S'_i) \wedge \neg\texttt{deact}(S'_i) \wedge \neg\texttt{maint}(S'_i)$

(6)+(9) -->

(11) $\texttt{rep\_active}(S_{i+1}) = \neg\texttt{v\_point}(S'_i) \wedge \texttt{active}(S'_i) \wedge \neg\texttt{deact}(S'_i) \wedge \neg\texttt{maint}(S'_i) \wedge \neg\texttt{repair}(S'_i)$

(6)+(10) -->

(12) $\texttt{rep\_active}(S_{i+1}) = \neg\texttt{v\_point}(S'_i) \wedge \texttt{act}(S'_i) \wedge \neg\texttt{deact}(S'_i) \wedge \neg\texttt{maint}(S'_i) \wedge \neg\texttt{repair}(S'_i)$

(9), (10), (11) and (12) occur from combining the previous rules. In fact they occur from the expansion of rule (4) and (6), which are expressed in terms of fluents over the current state, together with the rest rules. We do this in order to have a complete set of rules that represent the norm lifecycle and that are all expressed in terms of the previous state fluents. The complete set of rules to be modelled in the implementation (as we will see in the next section) will then be (1), (2), (3), (5), (7), (8), (9), (10), (11) and (12).

We adopt PDDL 2.1 [9] for our domain and problem representation. This, amongst others, introduces plan metrics, using numeric fluents, arithmetic operators and comparison predicates in the pre and post conditions of the actions. Plan metrics specify, for the benefit of the planner, the basis on which a plan will be evaluated (plan quality) for a particular problem. We use SGPlan 6 planner [12] which implements PDDL 2 and PDDL 3. SGPlan uses a modified Metric-FF planner for basic planning and optimises goal preferences.

6 Implementing the Example in PDDL

In this section we will show how our framework and the example scenario has been implemented in PDDL[3].

6.1 Domain and Problem Representation

As mentioned in Section 2 actions are accompanied by a utility function which represents the cost for the agent to execute the action, according to the state of the world when executed. In this scenario, we introduce a function parameter, the "total-cost", which represents the "cost" for the individual to execute it. The function change might vary according to the value of the fluents at every state, so it is not static in that it might increase or decrease depending on fluent formulas over the states. In our example, we assume that the cost to open the door is 1, to get out through the door in case there is no fire is 5 and in case there is fire is 50, to break the window is 5, to jump from the window is 1, and to pay the price of the fine is 10. In an alternative scenario, different values can be assumed (for example a higher price for the fine, say 100). The domain actions can be seen in Figure 2. The objective of the planner will be to try to minimize this cost while at the same time satisfying the conditions of Section 4.4.

[3] The reader is referenced to `http://www.lsi.upc.edu/~panagiotidi/pddl-escape` to find the detailed code for the example.

```
(define (domain escape1)
  (:requirements :adl :typing )
  (:types person - object)
  (:predicates
   (door-open) (on-fire)
   (out ?p - person)
   (window-broken) (fine-paid)
   (active-prohib1 ?p - person)
   (v-point-prohib1)
   (active-repair-prohib1 ?p - person)
   (checked)
  )
 (:functions (total-cost) )

 (:action open-door
    :parameters ()
    :precondition (and (not (door-open)) (not (checked)))
    :effect (and (door-open) (checked)
           (increase (total-cost) 1)))

 (:action go-out-from-door
    :parameters (?p - person)
    :precondition (and (door-open) (not (checked)))
    :effect (and (out ?p) (checked)
           (when (on-fire) (increase (total-cost) 50))
           (when (not (on-fire)) (increase (total-cost) 5))))

 (:action break-window
    :parameters ()
    :precondition (not (checked))
    :effect (and (window-broken) (checked)
           (increase (total-cost) 5)))

 (:action jump-from-window
    :parameters (?p -person)
    :precondition (and (not (out ?p)) (window-broken) (not (checked)))
    :effect (and (out ?p) (checked)
           (increase (total-cost) 1)))

  (:action pay_fine
    :parameters (?p -person)
    :precondition (and (out ?p) (not (checked)))
    :effect (and (fine-paid) (checked)
           (increase (total-cost) 10)))

;; NORM LIFECYCLE ACTION CODE
)
```

Fig. 2. Domain actions in PDDL

```
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;; act:     not out                           ;;
        ;; deact:   out                               ;;
        ;; maint:   not window-broken                 ;;
        ;; repair:  fine-paid                         ;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:action check-norms
  :parameters ()
  :precondition (not (checked))
  :effect (and (checked)
   ;;;;;;;;; ACTIVE ;;;;;;;; (1) (2);;;;;;;;;;;;;;
   (forall (?p - person)
    (when (and (active-prohib1 ?p) (not (out ?p)))
       (active-prohib1 ?p)))
   (forall (?p - person)
    (when (and (not (out ?p)) (not (out ?p)))
       (active-prohib1 ?p)))
   ;;;;;;;;;;;;;;;;;;; -ACTIVE (3) ;;;;;;;;;;;;;;;;;
   (forall (?p - person) (when (out ?p)
           (not (active-prohib1 ?p))))
   ;;;;;;;;;;;;;;;; VIOLATION POINT (9) (10);;;;;;;;;;;
   (forall (?p - person)
    (when (and (not (v-point-prohib1))
              (active-prohib1 ?p)
              (not (out ?p)) (window-broken))
               (v-point-prohib1)))
   (forall (?p - person)
    (when (and (not (v-point-prohib1))
            (not (out ?p)) (not (out ?p)) (window-broken))
                 (v-point-prohib1)))
   ;;;;;;;;;;;;;;; -VIOLATION POINT (5) ;;;;;;;;;;;;;
   (forall (?p - person) (when (v-point-prohib1)
               (not (v-point-prohib1))))
   ;;;;;;;;;;;; ACTIVE REPAIR NORM (7) (11) (12);;;;;;;;
   (forall (?p - person)
    (when (and (rep_active-prohib1 ?p) (not (fine-paid)))
              (rep_active-prohib1 ?p)))
   (forall (?p - person)
    (when (and (not (v-point-prohib1))
              (active-prohib1 ?p) (not (out ?p))
              (window-broken) (not (fine-paid)))
                  (rep_active-prohib1 ?p)))
   (forall (?p - person)
    (when (and (not (v-point-prohib1))
          (not (out ?p)) (not (out ?p)) (window-broken)
          (not (fine-paid))) (rep_active-prohib1 ?p)))
   ;;;;;;;;;;;;;; -ACTIVE REPAIR NORM (8);;;;;;;;;;
   (forall (?p - person) (when (fine-paid)
       (not (rep_active-prohib1 ?p))))     )
```

Fig. 3. Norm lifecycle representation in PDDL

```
(define (problem escape1)
  (:domain escape1)
  (:objects sergio - person)
  (:init
   (= (total-cost) 0)
   (on-fire))

  (:goal (and (out sergio)
    (forall (?p - person) (not (active-repair-prohib1 ?p)))
    (not (checked))) )

  (:metric minimize (total-cost)) )
```

Fig. 4. Example problem representation in PDDL

```
; Time 0.03
; ParsingTime 0.02
; NrActions 6
; MakeSpan
; MetricValue 16.000
; PlanningTechnique Modified-FF(best-first search) as the subplanner

0.001: (BREAK-WINDOW) [1]
1.002: (CHECK-NORMS) [1]
2.003: (JUMP-FROM-WINDOW SERGIO) [1]
3.004: (CHECK-NORMS) [1]
4.005: (PAY_FINE SERGIO) [1]
5.006: (CHECK-NORMS) [1]
```

Fig. 5. Plan result 1

```
; Time 0.03
; ParsingTime 0.02
; NrActions 4
; MakeSpan
; MetricValue 51.000
; PlanningTechnique Modified-FF(best-first search) as the subplanner

0.001: (OPEN-DOOR) [1]
1.002: (CHECK-NORMS) [1]
2.003: (GO-OUT-FROM-DOOR SERGIO) [1]
3.004: (CHECK-NORMS) [1]
```

Fig. 6. Plan result 2

Figure 3 depicts the PDDL code for the norm lifecycle. That is, the translation of the norm that prohibits the person to break a window while he is inside the building and in case he does, he needs to pay a fine. It consists of one action that implements the rules described in Section 5. At this point, it has to be pointed out that the

implementation supports norm instances. We consider every norm to possibly have several instances, in case it contains variables in its activation, deactivation and maintenance condition. In this case, the norm status needs to be checked for every possible instance of the norm.

Figure 4 depicts the PDDL code of the problem. The "total-cost" function is initialised to 0 and one person instance "sergio", who's objective is to be out of the building, is defined. The objectives described in Section 4.4 are directly reflected in the problem code. Since there is only one norm, a prohibition, the goal includes the requirement that all the repair instances of the norm are not active. Additionally, the planner is required to find a plan such that the "total-cost" is minimum.

6.2 Results

We experiment by changing the values of the action costs in order to produce different cases. We expect to see two different behaviours emerging from such changes, as explained in subsection 6.1. In case where the cost of breaking the window and paying the fine has a noticeable advantage (in terms of "total-cost") over the case where the individual gets out through the door, then the planner will pick the first plan, otherwise the second.

When running the experiment with the original values given in subsection 6.1 the planner produces the result depicted in Figure 5 with a total value of 16 (5 for breaking the window, 1 for jumping and 10 for paying the fine). By modifying the original cost of the action "pay_fine" to 100 instead of 10, we notice a change in the outcome of the planner execution as can be seen in Figure 6. This time the plan produced involves the actor getting out through the door with a total cost of 51 (1 for opening the door and 50 for getting out through the door). This can be intuitively explained, as the fine price has become too high compared to the toll the agent has to pay by getting out through the door. In the case the agent chose to break the window, jump and pay the fine, the "total-cost" would sum up to 106, which would be higher than getting out through the door.

Although this example is very simple, some preliminary test we have performed with other toy scenarios with more actions and norms showed that it scaled quite nicely. We foresee some scalability issues as we introduce more expressiveness in our framework (e.g. time constraints and time durations).

7 Related Work

There exists an wide research background in the understanding of how legal, or normative, systems are established within human societies, how they impact on the activities of social individuals and how they can be applied in electronic institutions [17,5,1]. An extensive analysis of such background is out of the aim of our paper, as most of the work concentrates on the theoretical aspects of the normative concepts from the societal perspective, and in those cases that the agent perspective is taken, little attention is made on how agents are able to take normative positions into account during their means-ends reasoning.

Close to our work is [4], where the authors create and extend a programming language that implements normative agents. They consider norms as being represented by counts-as rules and sanctions as rules of the opposite direction. In [3] the authors extend the previous by defining the properties enforcement and regimentation and a model checking component which verifies these properties. Although the approach seems promising, it focuses more in the deliberation step of the reasoning cycle.

In [13] the NoA system is presented. It comprises the NoA language for the specifications of plans, norms and contracts, and the NoA architecture, which operates as an interpreter and executor of such specifications and represents a concrete implementation of an approach to norm-governed practical reasoning. The plans get instantiated at runtime according to whether they can satisfy a norm and they are labelled as consistent or inconsistent according to the currently activated permissions and prohibitions. With this labelling mechanism, the deliberation process becomes informed about possible norm violations. However, the focus again is given to the deliberation step, not the means-ends reasoning, as plans are pre-computed.

The framework described by [8] shares some similarities with our approach. Their focus on sanctions (which, in our model, are implemented more via additional norms) means that they only allow for very specific, predefined normative states, and that violations in their framework may only occur once.

Our approach bears connection with the work presented in [16], in that we both endow norms with semantic features, using subsumption to check when a norm is triggered, when it is no longer active and so on. However, our work aims at connecting norms and plans, as plans provide pragmatic and realistic "histories" of computations (which we represent as sequences of states).

Sergot in [18,19] extends $C+$ by adding expressions of the form α "counts_as" β (this implying that every transition of type α counts also in specified circumstances as a transition of type β), calling the extended language $(C+)^{+}$. In addition, Sergot extends $C+$, calling the new language $(C+)^{++}$, by adding the "permitted" and "not-permitted" rules, implying in this way desired, legally permitted or not acceptable states and transitions. In this way it is possible to verify system properties that hold if all agents/system components behave in accordance with norms/social laws and to analyse system properties that hold when agents components fail to comply with norms. However the extensions provide no functional semantics to work with when it comes to realistic representation of norms and practical reasoning.

From a planning perspective our work is similar to PDDL 3.0 [10], an extension of planning language PDDL (originally implementing STRIPS) that imposes strong and soft constraints expressed in Linear Temporal Logic formulas on plan trajectories as well as strong and soft problem goals on a plan. Although we explored the option of using PDDL 3.0 soft constraints in our work, their expressiveness proved to be insufficient while trying to capture the semantics that we use for the norm lifecycle, mainly due to two reasons: 1)it lacks the operator "until", which would permit us to express the norm lifecycle (ex. a norm is violated when activated at some point and maintenance condition does not hold at some state after this and deactivating condition does not hold at any state in between) and 2)a norm can be activated and deactivated (and possibly violated) several times during the execution of a plan, something not possible to be expressed in PDDL 3.0.

In [14] the authors use a mechanism to choose a plan that will achieve individual and global goals while attempting to abide by a set of norms. They represent the environment affecting the agent as a transition system and the plans as Hierarchical Task Networks (HTN) [11] with the nodes specifying the actions that take place. They make use of a rule language to specify normative rules that identify the cases in which a norm starts, and ceases, to exist. Additionally, they adopt a utility based model of norm compliance. More specifically, they make the assumption that the execution of a plan results in some base utility, and that different types of norms are associated with different utility measures. They then create an algorithm that selects a path through the plan, and a set of norms (created by the rules as actions are executed) with which to comply, that is conflict free, and which lead to maximal utility. Conflicts are resolved by selecting actions where the cost of violating one set of norms is outweighed by the reward obtained in complying with another.

8 Conclusions

The work presented in this paper stems from the fact that little work on practical reasoning mechanisms within normative environments exists. Having this in mind, we have focused our attention on the practical introduction of norms in the plan generation process as a first step towards a fully normative-aware agent. Our approach is based on extending a normal STRIPS planning with a model of both the norms and their operational semantics. The result is that the planner generates optimal plans with respect to the norms (treating norms as soft restrictions) and the user preferences (modelled as costs) and leads to a reasoning mechanism that allows, given a set of norms, to create the most "precious" plan path during the trajectory of which each norm might or not be respected or violated and repaired (even more than one instances).

The strength of our normative framework lies on the realistic and fully functional representation of the normative reasoning problem. It uses semantics which have been implemented by several planners (STRIPS) while it only adds a small overhead to the planning process caused by the introduction of the intermediate states. We show how the normative concepts and the operational semantics covering norms' lifecycle have been represented in PDDL 2.1 and processed by a PDDL planner such as SGPlan 6.

Existing planners allow for several features to be integrated to the planning process (durative actions, etc) and we intend to expand the framework so that it includes time propositions and limitations within the definition of the norm. Our work can also be further extended towards utility functions over the states of the world instead of the actions. This alternative would allow for virtual representation of preferences of a state over another, which can be even closer to the way human reason about whether to follow the norm or not.

As mentioned in Section 6 no full analysis of the time overhead has been done. While experiments with small examples have shown no significant burden in the execution time, in larger planning domains we expect to see some cost. This is due to the fact that we force the planner to go through the intermediate states in order to calculate the norms' status. We noted further executional load in cases where norms might consist of several instances, and, the more the variable instances that relate to the norm instances in the problem file, the more severe the cost. We aim to do extensive studies on the

scalability of our approach in terms of the number of actions as well as the number of norms (and instances) that might be handled during the plan formation.

As mentioned in the introduction, our final aim is to model and implement the influence of norms in all the steps of the agents' reasoning cycle. To do that we plan to integrate our normative planner within a BDI agent implementation. Also we aim to include in our model more complex norm representations that contain full norms as repair norms (that might also contain their own repair norms).

References

1. Aldewereld, H.: Autonomy vs. conformity: An institutional perspective on norms and protocols. PhD Thesis, Utrecht University (2007), http://igitur-archive.library.uu.nl/dissertations/2007-0604-200758/UUindex.html
2. Alvarez-Napagao, S., Aldewereld, H., Vázquez-Salceda, J., Dignum, F.: Normative Monitoring: Semantics and Implementation. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN 2010 International Workshops. LNCS, vol. 6541, pp. 321–336. Springer, Heidelberg (2011)
3. Astefanoaei, L., Dastani, M., Meyer, J.J., de Boer, F.S.: On the Semantics and Verification of Normative Multi-Agent Systems. International Journal of Universal Computer Science 15(13), 2629–2652 (2009)
4. Dastani, M., Tinnemeier, N.A., Meyer, J.J.: A Programming Language for Normative Multi-Agent Systems. In: Multi-Agent Systems: Semantics and Dynamics of Organizational Models, Hershey, PA, USA (2009)
5. Dignum, F.: Autonomous agents with norms. Artificial Intelligence and Law (7), 69–79 (1999), http://www.springerlink.com/index/N32XU121L58H417V.pdf
6. Dignum, F., Broersen, J., Dignum, V., Meyer, J.-J.: Meeting the Deadline: Why, When and How. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 30–40. Springer, Heidelberg (2004)
7. Fikes, R., Nilsson, N.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2, 189–208 (1971)
8. Fornara, N., Colombetti, M.: Specifying and Enforcing Norms in Artificial Institutions. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 1–17. Springer, Heidelberg (2009)
9. Fox, M., Long, D.: PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains, pp. 1–48. University of Durham, UK (2009)
10. Gerevini, A., Long, D.: Plan constraints and preferences in PDDL3: The Language of the Fifth International Planning Competition. Department of Electronics for Automation, University of Brescia, Italy (2006)
11. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory & Practice, 1st edn. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann (May 2004)
12. Hsu, C., Wah, B.: The sgplan planning system in ipc-6. In: 6th Int. Planning Competition Booklet, ICAPS 2008 (2008)
13. Kollingbaum, M.J.: Norm-governed practical reasoning agents. University of Aberdeen (January 2005)
14. Oren, N., Vasconcelos, W., Meneguzzi, F., Luck, M.: Acting on Norm Constrained Plans. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 347–363. Springer, Heidelberg (2011)

15. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a Formalisation of Electronic Contracting Environments. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS, vol. 5428, pp. 156–171. Springer, Heidelberg (2009)
16. Şensoy, M., Norman, T.J., Vasconcelos, W.W., Sycara, K.: OWL-POLAR: Semantic Policies for Agent Reasoning. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 679–695. Springer, Heidelberg (2010)
17. Sergot, M.: The representation of law in computer programs: a survey and comparison, Tano. CompLex (Series), vol. 91(1), p. 99 (January 1991), http://books.google.com/books?id=8U-cAAAACAAJ&printsec=frontcover
18. Sergot, M.: An action language for modelling norms and institutions. SIKS Masterclass, Utrecht (2003), http://www.cs.uu.nl/jurix03/sergot.pdf
19. Sergot, M.J., Craven, R.: The Deontic Component of Action Language $nC+$. In: Goble, L., Meyer, J.-J.C. (eds.) DEON 2006. LNCS (LNAI), vol. 4048, pp. 222–237. Springer, Heidelberg (2006)

Towards Justifying Norm Compliance

Ioan Alfred Letia and Anca Goron

Technical University of Cluj-Napoca
Department of Computer Science
Baritiu 28, RO-400391 Cluj-Napoca, Romania

Abstract. We introduce justifications in normative systems with the aim to better understand the compliance to norms, including situations of conflict between norms that might arise. To stay at the level of explanations on justification we employ the Hybrid Justification Logic. We show the grounding of the abstract norms to the concrete ones within an example for a traffic management system. The interplay between norms is judged at the values of the society through Value-Based Argumentation. The justification for the way agents comply or not to the norms, relative to their social values, has been implemented in the CaSAPI argumentation environment.

1 Introduction

Norms should help societies of agents to achieve higher performance without restricting their autonomy too much. As systems evolve the need to accommodate different sets of norms requires a more refined understanding of their effects on the overall behavior of agents. Jones and Sergot [14] have shown the way on using a deontic and action logic to analyze a number of notions crucial to the understanding of organized interaction in institutions, starting with the following introduction.

> It is a commonplace feature of legal systems, and other norm-governed organizations, that particular agents are empowered to create certain types of states, by means of the performance of specified types of acts. Typically, the states created will have a normative character according to which obligations and rights are established for some agents vis-a-vis others, as for instance when a contract is made, or a marriage is effected, or ownership of an item is transferred. The performances by means of which these states are established will often be of a clearly prescribed, perhaps ritualized nature, involving the utterance of a particular form of words (e.g., the utterance of a specific type of performative sentence), or the production of a formal document, or the issuing of a pass, perhaps in a particular context (e.g., in the presence of witnesses).

We are concentrating here to develop a method that can help such systems visualize, in the sense of offering proofs, whether the norms have been complied with or broken.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 110–128, 2012.

Advantages and some of the problems when using norms in multi-agent systems have been shown lately to be of significant importance [20]. Controlling multi-party interactions with norms [7], but also programming norm change [19] and making norms concrete [1] have contributed to a deeper understanding of ways to achieve flexible systems and also to monitor them [2].

Provision of explanation generating functionality has been found to be very helpful for both developing and providing proofs when using ontologies [13] by way of explanations via justifications. We are pursuing a similar path for the usage of norms in multi-agent systems, looking to ways of exploiting a justification logic version [4,11,10] of a hybrid logic [3]. As remote positive/negative justification checkers we employ the value based argumentation [6,5].

2 Running Example

Example 1. We consider a Traffic Management System (TMS), responsible for controlling the traffic flow through an intersection. The system's functioning is based on a set of predefined abstract regulations or norms (table 1) that specify the permissions and obligations of the driver agents passing through the intersection. The role of each norm is to ensure the safety of the passing drivers, passengers and pedestrians, general traffic security and a continuous and regular flow.

Table 1. Set of norms for the Traffic Management System

Norm	Specification
N1	A driver must stop at the Red color of the traffic lights
N2	A driver may pass at the Green color of the traffic lights
N3	A driver must signal its intention of changing the lane or the driving direction
N4	A driver must not exceed the speed limit of 50 km/h when passing through the intersection

Any violation of the norms N1, ..., N4 is sanctioned by a fee and the application of 10 penalty points for the driver agent. When summing to 50 penalty points, the driving license of the driver is automatically suspended.

Having specified the set of abstract norms together with the corresponding sanctions to be applied in the case of infringement of one or more norms, we might assume that the proper functioning of the TMS is ensured. However, taking into consideration the examples offered by the real world, which is governed by unpredictable events, we can easily state that the above assumption is false. Nevertheless, we have to check if the set of abstract norms can cover all the possible concrete events inside the intersection.

In this sense, we consider the following scenario (figure 1). The driver of vehicle A stops on the right lane at the Red color of the traffic lights. Other three

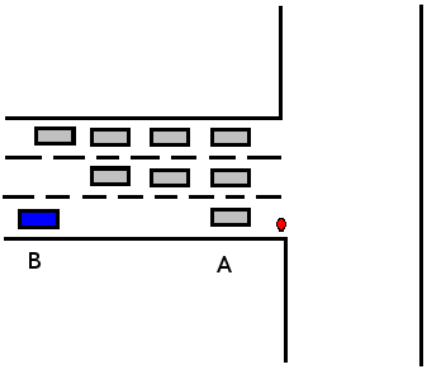


Fig. 1. Example scenario in traffic intersection

drivers stop on the middle lane and four drivers stop likewise on the left lane, waiting for the Green light of the traffic lights. In the meantime, an ambulance carrying a patient to the hospital (represented by vehicle B) approaches the intersection, signaling its presence for the other drivers.

Checking each of the three lanes, the driver of vehicle B decides to use the right lane to enter the intersection as there is only one vehicle stopped. The driver of vehicle A notices the approaching ambulance, checks its right and left side and decides to pass on the Red color of the traffic lights and drive to the right, clearing the right lane for the ambulance to pass. We might now raise the questions: Did the driver from vehicle A make the right choice? Did vehicle A violate the TMS norms? Should the driver from vehicle A be sanctioned for his actions?

For providing a proper answer for each of the above questions, we will try applying two reasoning methods and provide a justification for the course of actions chosen by the driver agent of vehicle A.

3 Fragment of the Hybrid Justification Logic

Justification logic [4] is based on the classical propositional logic augmented with justification assertions $t : F$, meaning that t is a justification for F. Hybrid logics are extensions of standard modal logics, involving symbols that name individual states in models [3]. The language of proofs we are using is a fragment of the justification logic of hybrid logics, that internalize both semantics and proofs [11,10].

In justification logic we can use *justification variables* $x_1, x_2, \ldots$ (capable of making relative justification conclusions) and *justification constants* $c_1, c_2, \ldots$. The operation $\cdot$ is an application operation, used to show that if t is the justification of $X \supset Y$ and u is the justification of X then $t \cdot u$ is a justification of Y.

Facts that make up a state of a possible world i should be certified by the facts f_i, the indexed family of constant symbols.

In the hybrid language $\mathcal{H}(@)$ the prefix operator $@_i$ is used for the formula $@_i\phi$, for saying that ϕ is true at the world named by i.

Definition 1. *Some of the operators of the basic hybrid justification logic [11] that we need in our presentation are the following.*

- **modus ponens**
$$\frac{X \;\; X \supset Y}{Y} \tag{1}$$
- $\cdot$ **axiom**
$$t : (X \supset Y) \supset (u : X \supset (t \cdot u) : Y) \tag{2}$$
- **remote fact checker***: For each nominal i and propositional letter P we have the facts*
$$@_i P \supset f_i : @_i P \tag{3}$$
which says that, in the possible world i, P is certified by the fact f_i that it is true, and
$$@_j \neg P \supset f_j : @_j \neg P \tag{4}$$
saying that, in the possible world j, the fact f_j certifies that $\neg P$ is true.
- **remote positive justification checker** *For any formula X justified by the term t and any nominal i the operator $!_i$ is used to show that t is checkable.*
$$@_i t : X \supset (!_i t) : @_i t : X \tag{5}$$
Here the operator $!_i$ plays the role of a certificate (e.g., as provided by some auditor) as proof that the formula has been proved to be true.
- **remote negative justification checker** *For any formula X justified by the term t and any nominal i the operator $?_i$ is used to show that t is not checkable.*
$$@_i \neg t : X \supset (?_i t) : @_i \neg t : X \tag{6}$$
The operator $?_i$ constitutes a kind of certificate saying that we have no evidence for the formula X to be true.

Example 2 (continued). If we represent the norm N1 by

$$redColor \supset stop \tag{7}$$

then we have the following derivation:

1. $@_1 redColor \supset f_{1red} : @_1 redColor$,
(remote fact checker, with the constant f_{1red} certifying that in situation 1 the redColor is true)
2. $c_{N1} : (redColor \supset stop) \supset (f_{1red} : @_1 redColor \supset (c_{N1} \cdot f_{1red}) : @_1 stop)$,
($\cdot$ axiom and (7), with the constant c_{N1} refering to the norm N1)

The justification for the action of the driver (stop) in the situation $@_1$, according to the norm N1 is given by the result of the above derivation which shows that due to the norm certified by $c_{N1} : (redColor \supset stop)$ and the fact of the certified color $f_{1red} : @_1 redColor$ they jointly proved $(c_{N1} \cdot f_{1red}) : @_1 stop$, that the driver was justified for the fact that it stopped.

Our running example shows how considering the law we have to comply to an abstract norm. But, apart from the abstract norm we also have to face their instantiation, that is the concrete norms, which have to take into account the diversity of the real world [1].

Normative systems should have an important role to play in various future implementations and deployment of such systems. But for realistic systems to be successful there should be some proof that in certain situations the actors involved have acted according to the norms or not. We have embarked in the study of ways to provide some proof for the behavior of agents and the Justification in the hybrid logic seems to be appropriate due to its power of expression.

4 Grounding the Norms

The Traffic Management System from our example represents a normative system, whose proper functioning is ensured by a pre-established set of norms (table 1). We can observe that each norm is defined in such a way to abstract from the concrete events and situations that it is supposed to cover [1], allowing this way to be applied for a wide range of situations and possible events.

However, the downside of using a high level of abstraction for norm specification is represented by the difficulty in relating abstract norms to concrete events that may occur inside the system. In order to overcome this issue, [1] proposes a solution based on the use of *counts − as* statements, which provide a link between institutional and concrete, real-world facts.

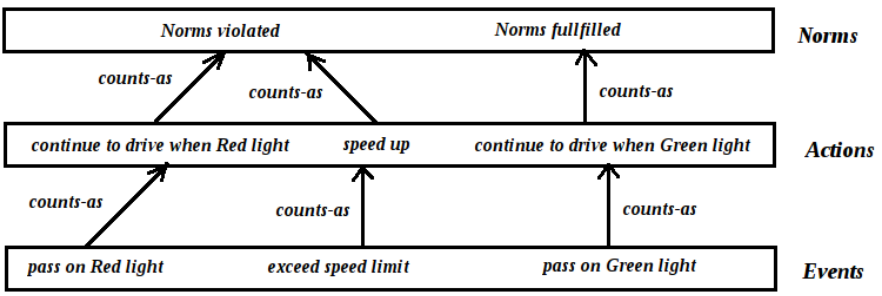


Fig. 2. Fulfillment and violation of TMS norms

Considering our scenario, *counts − as* statements can offer the possibility to represent the connection between concrete events and abstract concepts used for specifying the norms of the Traffic Management System under the form of static links, which highlight the connection between them, capturing the normative consequences of an agent's actions (figure 2). We will extend however the application of *counts − as* as described in [1] to capture also the connection between physical events and their abstracted non-instituional representation. By relating system events to agent actions, highlighting their effects, *counts − as* relations can provide a powerful reasoning means about the meaning of an agent action and what it brings about in a certain context, focusing in the same time on the normative impact [1] of performing those actions in order to achieve a certain goal.

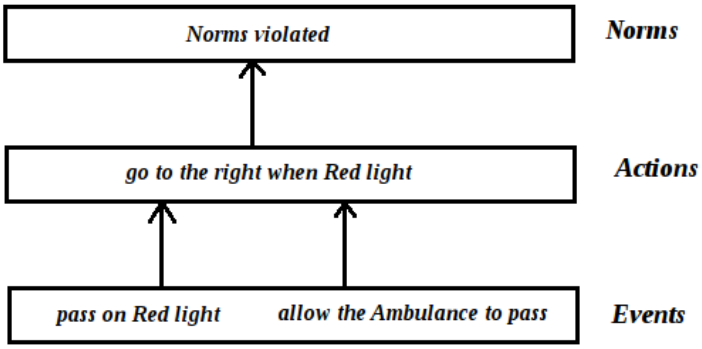


Fig. 3. Violation of TMS norms in a possible scenario

The relation between concrete or explicit actions and the events happening inside the intersection and the abstract norms governing the traffic flow is important to understanding how agents interpret the norms. We return now to the possible scenario highlighted in figure 1 and try to represent it based on counts-as statements, which prove useful when trying to emphasize what happens when the context changes (figure 3).

In the figure 3 the violation of norm N1 of the Traffic Management System by vehicle A is highlighted when passing on the Red color of the traffic lights and turning to the right for allowing the ambulance to pass. If we were to base our reasoning only on the TMS norms applied to this context, then we could easily conclude that the agent made a wrong decision and he must be sanctioned for its actions.

However, if we extend the set of norms on which our reasoning is based and include also a more general driving norm, more specifically the norm stating the obligation of each driver to allow emergency vehicles (police cars, ambulances

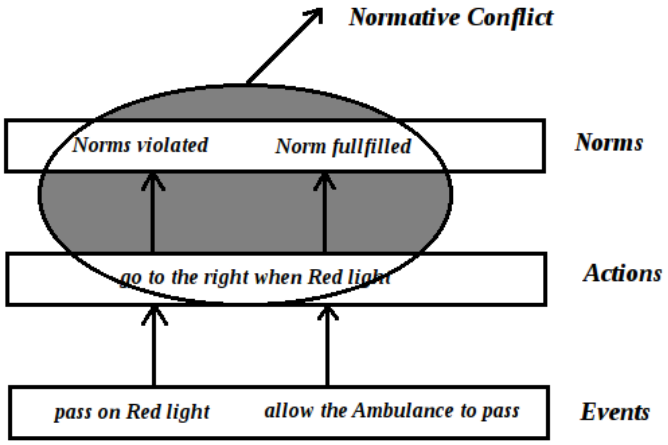


Fig. 4. Conflict between TMS and general applying norms

etc.) to pass, moving out of their way, then we can no longer easily arrive to the conclusion that the agent made the right or the wrong choice. The new representation is captured in figure 4, while the extended set of norms is emphasized in table 2.

Table 2. Extended set of norms for the Traffic Management System

Norm	Specification
N1	A driver must stop at the Red color of the traffic lights
N2	A driver may pass at the Green color of the traffic lights
N3	A driver must signal its intention to change lane or driving direction
N4	A driver must not exceed the speed limit of 50 km/h when passing through the intersection
N5	A driver must allow emergency vehicles to pass by moving out of their way

By analyzing figure 4, we can observe that a conflict is generated between norms N1 and N5 based on the fact that the same action of passing on the Red color of the traffic lights leads to both the violation and the fulfillment of the TMS norms.

Example 3 (continued). If we represent the norm N5 by

$$emergency \supset \neg stop \tag{8}$$

then we have

1. $@_2 emergency \supset f_{2emerg} : @_2 emergency$,
   (remote fact checker showing that 2 was an emergency situation)
2. $@_2 \neg c_{2stop} : stop \supset (?_2 c_{2stop}) : @_2 \neg c_{2stop} : stop$),
   (the remote negative justification checker, showing that no constant c_{2stop} is able to certify that the driver did stop on the way of the ambulance in the situation 2, where norm N5 (8) applies)

showing that the driver did ¬stop, thereby allowing the ambulance to pass. The fact that we have no proof that the driver did stop is obtained from some monitor [1] (video, witness, etc.), a trusted source.

5 Value-Based Argumentation for Compliance

If some of the remote justification checkers, needed for justification, can be obtained from trusted monitors, in more complicated real situations we need to also consider the values supported by the specific agent society. For our running examples these can be *traffic_security*, *pedestrian_safety*, etc.

Definition 2. *[5] A value based argumentation framework (VAF) is a 5-tuple: VAF {AR, attacks, V, val, P}, where AR represents a finite set of arguments, attacks is an irreflexive relation on the set AR, V is a finite nonempty set of values, val is a function mapping the elements of AR to the elements of V and P is a set of possible audiences. Moreover, an argument A relates to value v if accepting A promotes or defends v, where v = val(A), val(A) ∈ V, for every A ∈ AR.*

By allowing ordering on the values to be applied [15] extends the VAF, so that an argument $A \in AR$ successfully attacks (defeats) an argument $B \in AR$ only if the value promoted by B is not ranked higher than the one promoted by A, according to some partial ordering $>$ of values. Furthermore, the notion of an argument promoting or demoting values to a given degree X (denoting a real number) is introduced ($val = (A, X, V, P)$), meaning that if A and B would promote the same value v, A would defeat B if it promotes the value v to a higher degree. In practical reasoning, arguments are used to provide a presumptive justification for a certain action, which instantiates an argument scheme, representing an extension of the Walton's sufficient condition scheme [6]:

In the current circumstances R
We should perform action A
Which will result in new circumstances S
Which will realize goal G
Which will promote value V.

If we consider again the Traffic Management System example, we can observe that, similarly to the real world, norms are defined to ensure values, such as traffic security, pedestrians safety, efficiency and flow control, by their fulfillment.

The violation of one or more norms brings about insecurity and traffic disturbances as the mentioned values are no longer supported. A special case is represented by norm N5 stating the obligation of each driver to allow emergency vehicles to pass, a norm whose fulfillment brings about values such as life for our example in which the driver agent must allow an ambulance carrying a patient in critical condition to pass. If norm N5 is violated, the life of the patient is in danger as it may delay the arrival to the hospital and his condition might aggravate.

Table 3. Arguments for justifying the actions of the driving agent relative to our values, and in current circumstances

Arg.	Argument Specification	Promoted/Demoted
A_1	Should wait for the Green light before driving to the right. Does not affect traffic in intersection, and promotes security, safety and flow.	+traffic_security +flow_control +pedestrians_safety -life
A_2	Should climb on right sidewalk. Allows ambulance to pass, promotes life, flow and traffic security	-pedestrians_safety +flow_control +traffic_security +life
A_3	Should not climb on right sidewalk, but wait for the Green light. May not create enough space for ambulance, may injure pedestrians, promotes safety.	+pedestrians_safety +flow_control +traffic_security -life
A_4	Should try moving on middle lane in front of the first car, and waiting for traffic lights, allowing ambulance. Promotes life and pedestrians safety.	-traffic_security -flow_control +life +pedestrians_safety
A_5	Should not move on middle lane, but wait for the Green light. May not be enough space in front of first car. Promotes security, flow, pedestrian safety.	+traffic_security +flow_control +pedestrians_safety -life
A_6	Should drive to right and pass on the Red light allowing ambulance, signaling and slowing down. Promotes life and safety.	-traffic_security -flow_control +life +pedestrians_safety

Each of the previously mentioned values will be further considered in the reasoning process based on the value-based argumentation [9] for what would be the right solution to apply in such a context and, based on the outcome, to provide a positive or negative justification for the choice of the driving agent. If we were to take into consideration only those actions that fulfill the initial TMS set of norms as the right choices, thus actions that promote values such as traffic security, flow control and pedestrians safety, then the agent should choose to stop at the Red color of the traffic lights. If we consider that the life of the patient is of higher importance, then the choice will be totally different.

We notice that the selection of the right action depends merely on the importance of promoting a certain value over another. Thus, as a first step we must decide over a hierarchy of values ranked according to their importance. In this direction, we apply an ordering relation α over the set of values V={*pedestrians_safety*, *traffic_security*, *flow_control*, *life*}, obtaining the following hierarchy of values: *flow_control* < *traffic_security* < *pedestrians_safety* < *life*, where the value of flow control has the lowest priority in the reasoning process and the value of life the highest priority. Moreover, the entire reasoning process will be translated from the normative level to the agent actions level, which comprises also the promoted or demoted values by each action. The arguments used for justifying each of the possible options are presented in table 3.

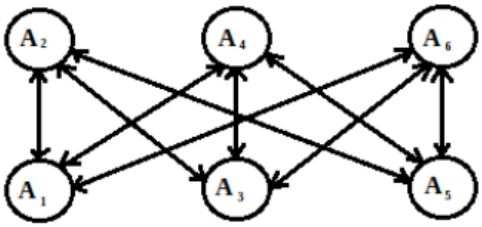


Fig. 5. Dung argumentation scheme for the traffic system

If we consider the Dung abstract argumentation scheme and the fact that only the actions that promote the value of *life* are to be chosen according to the α ordering relation, then any pair of arguments that do not support it are conflicting. Figure 5 summarizes the different conflicts between arguments. The system has two maximal sets A_2, A_4, A_6 and A_1, A_3, A_5, the first one encapsulating all the options that promote the value of life, while the second one those that demote that value. Thus, the actions suggested by arguments A_2, A_4 and A_6 are equally preferred in a Dung system.

As can be seen from figure 5, the Dung argumentation scheme is not enough for making decisions as we cannot establish which option promoting the value of $life$ is the best one to be selected. Therefore, we will extend the above presented scheme and consider also the strength of the arguments in decision making. In this way we can reduce the number of attack relations considering the fact that an attack may fail if the attacked argument is stronger than the attacker.

Therefore, we will consider as a stronger argument the one that promotes, besides the value of *life*, the maximum number of values immediately following *life* in the ordering relation α. The set of admissible arguments will be reduced in this case to the three A_2, A_4, A_6. The new attack relations are: $A_6 \rightarrow A_2$ and $A_4 \rightarrow A_2$, both arguments A_6 and A_4 successfully attacking A_2, as they promote

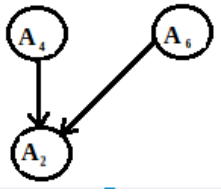


Fig. 6. Reduced argumentation scheme for the traffic system

the value of *pedestrians_safety*, the second in the value ranking, while argument A_2 promotes the values of traffic security and flow control, both of them being lower ranked values.

In the reduced argumentation scheme presented in figure 6, both arguments A_6 and A_4 successfully attack A_2. However, a new criterion must be considered in order to decide upon the best action to take from the ones suggested by the two mentioned arguments. In this direction, we consider an extension of the argumentation framework based on contextual preferences. For the case here, we can observe that we have two arguments that promote the same values. In order to decide which of the two arguments is stronger, we must consider also other aspects strictly applicable to the context in which the decision must be taken. Regarding our example, if we consider that the middle lane is free, then vehicle A can move to that lane allowing the ambulance to pass and promoting the value of *life* without the need to violate norm 5, thus promoting *life* at a greater extent than argument A_6.

However, if we consider the situation in which the middle lane is not free, then vehicle A cannot change lane safely and therefore the ambulance may not be able to pass. Therefore for this context, argument A_6 promotes *life* at a greater extent. Considering the contextual-based preference argumentation framework (CPAF) definition from [8], we define the set **C** of contexts as containing two different contexts: c_1, referring to the situation in which the middle lane is free and c_2 referring to the situation in which cars are stopped on the middle lane. If we consider context c_1, then argument A_4 will successfully attack argument A_6: $A_4 \rightarrow A_6$, while for context c_2: $A_6 \rightarrow A_4$.

Considering the likelihood for contexts c_1 and c_2, we can state that the likelihood for context c_2 has a higher probability than the one for context c_1, as it is more likely for cars to be stopped also on the middle lane in an intersection than for the middle lane to be free. Finally, we can say that argument A_6 successfully attacks argument A_2 and therefore the right solution for our agent is to pass on the Red color of the traffic lights and turn to the right allowing for the ambulance to pass without jeopardizing the life of the patient.

However, we also have to consider the negative aspect of his choice, that is the risk of jeopardizing the traffic flow through the intersection and the safety and security of the other agents.

Moreover, we can observe that using a VAF based reasoning approach in such a situation provides us with the possibility to decide upon the course of actions to follow, and, additionally, it offers us a viable justification [9] for the infringement of one or more norms in order to achieve a goal and promoting a higher ranking value.

Example 4 (continued). Here the assumption is that we do not have norm N5, but we can consider an argument like A6 that promotes the value life.

$$emergency \supset (\neg stop \supset promotesLife) \tag{9}$$

Then we have

1. $@_2 emergency \supset f_{2emerg} : @_2 emergency$,
   (remote fact checker)
2. $@_2 emergency \supset @_2(\neg stop \supset promotesLife)$,
   (modus ponens axiom (1))
3. $c_{2stop} : @_2 \neg stop \supset (!_2 c_{2stop}) : @_2(c_{2stop} : \neg stop))$,
   (remote positive justification checker with the constant c_{2stop} certifying that the driver did not stop in situation 2)
4. $f_{2emerg} : (c_{A6} : (\neg stop \supset promotesLife) \supset (c_{2stop} : @_2 \neg stop)) \supset (f_{2emerg} \cdot c_{2stop} \cdot c_{A6}) : @_2 promotesLife$,
   (applying axiom (2) with argument A6 given by (9))

showing that the driver did not stop on Red light, promoting life (A6).

For the given situation $@_2$ where we have an emergency, we have the justification of the fact f_{2emerg} to apply argument A6 for $\neg stop \supset promotesLife$ with the constant justifier c_{A6}. Therefore, the justification that proves that the driver acted to promote life, is expressed by the term $f_{2emerg} \cdot c_{2stop} \cdot c_{A6}$ showing that in the situation referred to we indeed have $@_2 promotesLife$.

6 Implementation of Norm Justification

The reasoning process presented in the previous section has been implemented in the CaSAPI [12] argumentation environment[1], based on the MARGO platform[2] for practical reasoning. MARGO uses a logic language $\mathcal{L}$ to represent knowledge, goals, and decisions of an agent, allowing to assign to each of these items different priorities according to knowledge probabilities, preferences or decision utilities.

Based on this information and on an assumption-based argumentation framework to which a general argumentation framework is translated [17] , MARGO evaluates and suggests decisions, justifying at the same time the choice made.

For the implementation of the decision making process in the traffic system, first a main goal is established for our agent and that is to have the right reaction in that situation, followed by the definition of the three sets of rules on which the decision process is based [17]:

[1] Available at `http://www.doc.ic.ac.uk/~dg00/casapi.html`
[2] Available at `http://margo.sourceforge.net/`

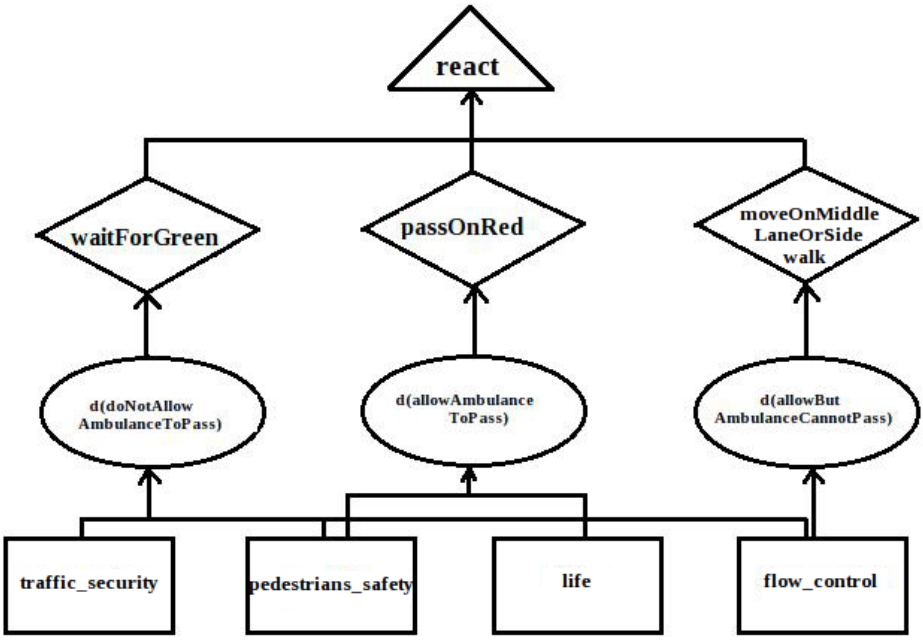


Fig. 7. Agent dilemma representation in MARGO

- goal rules: R: $G_0 \leftarrow G_1, ..., G_n$, where $n > 0$ and G_i is a goal literal in $\mathcal{L}$, meaning that the abstract goal in the head of the rule is promoted or demoted by the goals in the body.
- epistemic rules: R: $B_0 \leftarrow B_1, ..., B_n$, where $n \geqslant 0$ and B_i is a belief literal in $\mathcal{L}$, meaning that B_0 is true if the conditions $B_1, ..., B_n$ are satisfied.
- decision rules: R: $G \leftarrow D(a), B_1, ..., B_n$, where $n \geqslant 0$ and $D(a)$ is a decision literal in $\mathcal{L}$, meaning that the concrete goal G is promoted or demoted by the decision $D(a)$ with the condition that $B_1, ..., B_n$ are satisfied.

For our example, the goal is divied into three subgoals:

- $G_1 \leftarrow waitForGreen(X)$
- $G_2 \leftarrow passOnRed(X)$
- $G_3 \leftarrow moveAside(X)$,

each of which representing a possible solution corresponding to three possible decisions:

- $d(doNotAllowAmbulanceToPass(X))$
- $d(allowAmbulanceToPass(X))$
- $d(allowButAmbulanceCannotPass(X))$

According to the goal theory, the achievement of the main goal is done by deciding on one of the three possible options. However, the moral dilemma comes from deciding upon the proper reaction in such a context. Therefore, the selection must take into consideration also the values brought about by performing a certain action: *traffic_security*, *flow_control*, *pedestrians_safety* and *life*, with the implementation of the reasoning process shown in figure 8.

```
%Goal theory
goalrule(r01, react(X), [waitForGreen(X)]).
goalrule(r02, react(X), [passOnRed(X)]).
goalrule(r03, react(X), [moveOnMiddleLaneOrSidewalk(X)]).

%Decision theory
decisionrule(r10(X), waitForGreen(X), [d(X),
      doNotAllowAmbulanceToPass(X),
      traffic_security(X), pedestrians_safety(X),
      flow_control(X), sn(life(X))]).
%life
decisionrule(r20(X), passOnRed(X), [d(X),
      allowAmbulanceToPass(X),
      sn(traffic_security(X)), pedestrians_safety(X),
      sn(flow_control(X)), life(X)]).
%flow control
decisionrule(r30(X), moveOnMiddleLaneOrSidewalk(X), [d(X),
      allowButAmbulanceCannotPass(X), sn(traffic_security(X)),
      sn(pedestrians_safety(X)), flow_control(X),
      sn(life(X))]).

%Priorities
decisionpriority(r20(X), r30(X)).
decisionpriority(r20(X), r10(X)).

%Epistemic theory
epistemicrule(f00, doNotAllowAmbulanceToPass(traffic_participants),[]).
epistemicrule(f01, doNotAllowAmbulanceToPass(patient),[]).
epistemicrule(f02, allowAmbulanceToPass(traffic_participants),[]).
epistemicrule(f03, allowAmbulanceToPass(patient), []).
epistemicrule(f04, ambulanceCannotPass(traffic_participants),[]).
epistemicrule(f05, ambulanceCannotPass(patient),[]).
    ...
decision([d(allowAmbulanceToPass(patient)),
    d(doNotAllowAmbulanceToPass(traffic_participants)]).
    assumable(life(patient)).
    ...
```

Fig. 8. CaSAPI description of reasoning process

The basic sets of rules are further represented for our example. First, the set of goal rules $\{passOnRed(X)$, $waitForGreen(X)$, $moveOnMiddleLaneOrSide-walk(X)\}$ followed by the decision rules, establishing the implied actions and the promoted (and/or demoted) values. Each decision rule corresponds to one argument from the argumentation scheme (table 3) as follows: the first decision rule $r10$ represents the option of waiting for the Green light, corresponding to argument A_1, which promotes $traffic\_security(X)$, $pedestrians\_safety(X)$, $flow\_control(X)$ and demotes $\neg life$ $(sn(life(X))$, which is represented by applying a strong negation on the value of *life*. Similarly, the second decision rule $r20$ corresponds to argument A_6, promoting the values of life $(life(X))$ and pedestrians safety, while rule $r30$ corresponds to either of the arguments A_2 and A_4.

Considering the α ordering relation, the achievement of the main goal is done by choosing an action that promotes the value of life, therefore the priority rules:

- The decision of passing on Red has a higher priority than the decision of waiting for green
- The decision of passing on Red has a higher priority than the decision of moving aside

According to the epistemic theory [17], the agent has conflicting beliefs about which decision represents the right one and whether by selecting one option the security of the traffic participants will not be affected or whether the life of the patient inside the ambulance will be jeopardized. The agent does not know for sure the exact gravity of the condition of the patient carried by the ambulance and therefore considering the life of the patient in jeopardy by one of his actions represents in this case an assumption. Taking into consideration all these aspects, six epistemic rules are specified, one for each different belief of the agent about the possible actions ($doNotAllowAmbulanceToPass$, $allowAmbulanceToPass$, $ambulanceCannotPass$) to perform for protecting either the patient (X=$patient$) or the traffic participants (X=$traffic\_participants$).

```
admissibleArgument(react(patient),P,S).
SENT= [d(patient), wn(del(f03)), wn(del(f13)),
      wn(del(f23)), wn(del(f33)), wn(del(f41)),
      wn(del(r02)), wn(del(r20(patient)))]
P = [passOnRed(patient)],
S = [d(allowAmbulanceToPass(patient))]
```

Fig. 9. Solution considering the best interest of the patient

After the knowledge description and representation of possible decisions, different admissible arguments are generated and solutions are suggested taking into consideration the best interest of the patient from the ambulance on one side (figure 9) and that of the traffic participants on the other side (figure 10).

```
admissibleArgument(react(traffic_participants),P,S).
SENT= [d(traffic_participants), wn(del(f00)), wn(del(f10)),
      wn(del(f20)), wn(del(f30)), wn(del(f42)), wn(del(r01)),
      wn(del(r10(traffic_participants)))]
P = [waitForGreen(traffic_participants)],
S = [d(doNotAllowAmbulanceToPass(traffic_participants))]
```

Fig. 10. Solution considering the best interests of the traffic participants

Figure 9 displays the result returned by the admissibleArgument(react (patient),P,S) predicate, result referring to the agent reaction when considering the best interest of the patient. P can be understood as the strongest combination of sub-goals which can be reached by an alternative. This sub-goal can be challenged. In this direction, admissibleArgument(passOnRed(patient),P,S) returns $P = [allowAmbulanceToPass\ (patient)]$, and
$S = [d(allowAmbulanceToPass(patient))]$.

Since this argument is a sub-argument of the previous one, they suggest the same alternatives. Similarly, figure 10 displays the admissible argument sustaining the agent reaction when considering the best interest of all traffic participants. In this case, the admissible argument suggests the action of waiting for the Green color of the traffic lights and thus not allowing the ambulance to pass.

We can observe that the solution proposed when considering the best interest of the patient as the main criterion ($X = patient$) is that of passing on the Red color of the traffic lights to allow this way the ambulance to enter the intersection (figure 9), while choosing to wait for the Green color of the traffic lights and not allowing the ambulance to pass is considered to be the best option when selecting as main criterion the best interest of the traffic participants ($X = traffic\_participants$) (figure 10).

7 Related Work

Vasconcelos et al. [20] define conflicts as situations that arise when an action is simultaneously prohibited and permitted/obliged, and its variables have overlapping values. The variables of a norm specify its scope of influence, that is, which agent/role the norm concerns, and which values of the action it addresses. Moreover, they propose a normative conflict resolution mechanism based on the addition of constraints that will prevent variables from having overlapping values. However, the proposed solution restricts the norm representation selection to the use of atomic formulas and might prove difficult or too complex to be applied in different situations as, for example, when dealing with deontic conflicts, that is when an action may be simultaneously prohibited and obliged. As an alternative solution to the normative conflict resolution problem, we further propose a method based on the reasoning power of value based argumentation.

Making norms concrete [1] relate the abstract concepts used in the specification to concrete ones used in practice. The assumption is that the "counts-as"

mechanism will also take care of the conflicts that might appear between the concrete norms. This could be the case in simple organizations, but in more realistic ones, due to the complexity of the real world conflicts are bound to appear. It is in such cases that our approach, showing a justification for how norms are being complied to, could be found useful.

Considering how power may be used by agents to govern the imposition of norms and the management of norms [18], it is also important to justify why and how in some specific situations the agents have behaved in a particular manner. Although having agents that can play certain roles and are able to apply power by changing the system's norms are quite natural, it is also necessary for the agents to understand how the norms are being enforced in the running of such a society. Our contribution provides a justification of how norms, that may conflict in some situation, are being interpreted by the agents for the values of the society.

8 Discussion and Conclusions

In the conclusions of their paper [16], the authors present the following situation regarding the requirements for the deployment of normative systems.

> Providing explanations of norm violations is important if managers are to appropriately assign responsibility and apply sanctions. Explanations can also help to evolve the normative specification of a system in order to prevent future violations. Thus far, monitors provide limited explanation of violation and fulfillment, in terms of the labels of the arcs transitioned. Future work will investigate generation of more comprehensive explanations. This may require reference to representations of work-flow separate from that implicitly specified by norms. This information may in turn be gleaned from observers.

Our aim in this paper has been to provide a method that can show in a concise and precise manner whether some norms have been complied to or broken.

Mechanisms for the detection and resolution of normative conflicts [20] to avoid them at design time are very important, but we have focused here on proofs for the justifications of agent behavior, by allowing the agents to make the proper decision considering the current environmental situation. Work on adequately dealing with unpredictable and dynamic environments where normative frameworks are deployed, by providing mechanisms for modifying the norms at runtime [19] obviously require understanding of the workings of norms. This is a significant aspect since otherwise the developers cannot know exactly how the norms should be changed.

Although we have used a very simple illustrative example, we are aiming at a general behavior of collaborating/competing agents, including interaction in normative organizations [7].

Considering the formalism for the monitoring of both regulative and substantive norms using monitoring [2] we feel that our approach can bring some light in this direction. One issue that has been raised during the presentation

at the workshop concerned the monotonicity of the hybrid justification logic. For justifying just some aspects along with the behavior of agents, in a kind of postmortem check, the lack of non-monotonicity should not be a problem.

Acknowledgments. We are very grateful to the anonymous reviewers for their useful comments that lead to a significant improvement of the paper. Part of this work was supported by the grant ID_170/672 from the National Research Council of the Romanian Ministry for Education and Research.

References

1. Aldewereld, H., Alvarez-Napagao, S., Dignum, F., Vazquez-Salceda, J.: Making norms concrete. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, vol. 1, pp. 807–814. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2010)
2. Alvarez-Napagao, S., Aldewereld, H., Vázquez-Salceda, J., Dignum, F.: Normative Monitoring: Semantics and Implementation. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN 2010 International Workshops. LNCS, vol. 6541, pp. 321–336. Springer, Heidelberg (2011)
3. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 821–868. Elsevier, Amsterdam (2007)
4. Artemov, S.: The logic of justification. The Review of Symbolic Logic 1, 477–513 (2008)
5. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation 13, 429–448 (2003)
6. Bench-Capon, T., Atkinson, K., McBurney, P.: Altruism and agents: an argumentation based approach to designing agent decision mechanisms. In: Decker, K.S., Sichman, J.S., Sierra, C., Castelfranchi, C. (eds.) Proceedings of the Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2009), Budapest, Hungary, pp. 1073–1080 (2009)
7. Boissier, O., Balbo, F., Badeig, F.: Controlling Multi-party Interaction within Normative Multi-agent Organizations. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN 2010 International Workshops. LNCS, vol. 6541, pp. 357–376. Springer, Heidelberg (2011)
8. Bourguet, J.-R., Amgoud, L., Thomopoulos, R.: Towards a Unified Model of Preference-Based Argumentation. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 326–344. Springer, Heidelberg (2010)
9. Burgemeestre, B., Hulstijn, J., Tan, Y.-H.: Value-Based Argumentation for Justifying Compliance. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 214–228. Springer, Heidelberg (2010)
10. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132, 277–322 (2005)
11. Fitting, M.: Justification logics and hybrid logics. Journal of Applied Logic 8, 356–370 (2010)
12. Gaertner, D., Rodríguez-Aguilar, J.A., Toni, F.: Agreeing on Institutional Goals for Multi-agent Societies. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS, vol. 5428, pp. 1–16. Springer, Heidelberg (2009)

13. Horridge, M., Parsia, B., Sattler, U.: Justification Oriented Proofs in OWL. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 354–369. Springer, Heidelberg (2010)
14. Jones, A.J.I., Sergot, M.: A formal characterisation of institutionalised power. J. of the IGPL 4, 427–443 (1996)
15. Modgil, S.: Value based argumentation in hierarchical argumentation frameworks. In: Conference on Computational Models of Argument, pp. 297–308. IOS Press (2006)
16. Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009, vol. 1, pp. 153–160. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2009)
17. Morge, M.: An argumentation-based computational model of trust for negotiation. In: Proc. of the International Symposium on Behaviour Regulation in Multi-Agent Systems, vol. 4, pp. 31–36 (2008)
18. Oren, N., Luck, M., Miles, S.: A model of normative power. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, vol. 1, pp. 815–822. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2010)
19. Tinnemeier, N., Dastani, M., Meyer, J.J.: Programming norm change. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, vol. 1, pp. 957–964. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2010)
20. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Normative conflict resolution in multi-agent systems. Autonomous Agents and Multi-Agent Systems 19, 124–152 (2009)

Normative Run-Time Reasoning for Institutionally-Situated BDI Agents

Tina Balke[1,2], Marina De Vos[2], and Julian Padget[2]

[1] University of Surrey, Centre for Research in Social Simulation
t.balke@surrey.ac.uk
[2] University of Bath, Dept. of Computer Science
{mdv,jap}@cs.bath.ac.uk

Abstract. Institutions, also referred to as normative systems, offer a means to govern open systems, in particular open multi-agent systems. Research in logics, and subsequently tools, has led to support for the specification, verification and enactment of institutions. Most effort to date has focused on the design-time properties of institutions (either on the normative or the system level), such as whether a particular state of affairs is reachable or not from a given set of initial conditions. Such models are useful in forcing the designer to state their intentions precisely, and for testing (design-time) properties. However, we identify two problems in the direct utilization of design-time models in the governance of live (run-time) systems: (i) over-specification of constraints on agent autonomy and (ii) generation of design-time model artefacts. In this paper we present a methodology to tackle these two problems and extract run-time reasoning components from a design-time model. We demonstrate how to derive an event-based run-time model of institutions that can be incorporated into the capabilities of autonomous BDI agents to address the issues above in order to realize practical norm-governed multi-agent systems.

1 Introduction

The motivation for this work is two-fold: the first is the goal of the run-time governance of open distributed systems and the second is a case study of such a system: using institutions to govern the interaction of participants to demonstrate the economic viability of future mobile phone networks (called wireless grids)

The general idea in these future mobile phone networks is that mobiles dynamically construct ad-hoc wireless grids with the objective of achieving (i) faster download times by splitting content into parts, downloading some using 3G and acquiring the rest from nearby phones using wifi (ii) reducing power consumption by trading off high-cost 3G communication for low-cost wifi communication [7].

The main challenge within this context is how to encourage participants to contribute actively to the collective by downloading their share of parts via the high-cost 3G link and then to share them with the others, rather than free-ride by only receiving parts from the other participants in the system.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 129–148, 2012.

We propose an agent-based simulation to analyse cooperation effects in these mobile phone networks, using institutions as a mechanism to encourage and enforce cooperation. Results from the simulation can then be used to determine the viability of the system.

Typically institutions and agents co-exist in a state of tension: agents are (supposed to be) autonomous, while institutions constrain autonomy. Often, in norm-governed MAS, this tension is alleviated by regimenting agents and their actions [12], thus not allowing any norm-deviation. We, in contrast, use a more social form of institution. An agent can query institutional properties at run-time in order to examine how situations were achieved, or can possibly be achieved in the future, to determine which normative context is applicable to their current situation and to evaluate possible futures and make a decision on being norm-compliant or not, on the basis of this information.

Given the event-based nature of the simulation, the institutional approach described in [3] offers a suitably compatible model, as well as a complementary computational model that could be adapted to provide agents with information about the institutional state. At the same time, we also needed a suitable agent architecture, with a programming model that would fit the requirements for both being able to process institutional events and taking a goal-driven approach to the tasks to be fulfilled in the simulation.

Traditionally, when trying to analyse normative effects on a system, the real world is formalized as two *separate* (i.e. no dependencies) models: a system model and a normative/institutional model of which only the design-time properties are analysed *separately*. While useful, this can be problematic when wanting to analyse the interplay between agents and the institution. Furthermore, it poses the problem of how and when to account for run-time effects. Thus, in contrast to the separate analysis of the normative and system models, we are interested in an integrative and coherent analysis of the two models indvidually and jointly. Thus, we approach the institutional and system modelling in two phases:

1. *Design-Time Model:* we start traditionally with a design-time model, which in contrast to the standard approaches, integrates the system and normative model and allows for design-time reasoning about the interplay between agents on the system level and the institutions on the normative level. For instance, we build an institutional model of the wireless grid concept to evaluate whether it makes sense to pursue the idea, i.e. whether enforcement improves collaboration even in the most favourable circumstances. A design-time model is less labour intensive than implementing a full system. This model hard-codes simplifications of the environment in which the agents interact, but it can be used for validation purposes and helps to expose requirements issues.
2. *Run-Time Model:* Using the design-time model as a starting point, we then derive a run-time model. This is created by removing all but the normative information and domain facts from the design-time model, in order to avoid design artifacts and restrictions with respect to agent autonomy.

> Instead of using the formal model on the system level, we employ a multi-agent simulation that is linked with the normative model. The run-time model provides the (BDI) agents in the simulation or the MAS system with a kind of oracle, that can respond to queries both about the current state and the normative consequences of actions.

The experience gained during the development and execution of these two phases leads to the main contribution of this paper: a *methodology* for developing design-time and run-time institutional models—that is, models that play a key part in *the development and the running of* either an application or, as in our case, a simulation, and expressing the rules of governance of an open system. In this respect, the simulation and its results are tangential to the main contribution, which is normative design and making such models accessible to agents.

The remainder of the paper is laid out as follows: in the next section (Section 2) we briefly describe the wireless grid scenario to provide the problem context. The background on norm-governed systems and the normative model adopted is provided in Section 3. In Section 4 we contrast the design and run-time model to highlight the different features of each and put forward some design rules to observe in deriving the run-time model from the design-time one. This is followed by a presentation of the both models for our scenario and a description how the normative state is monitored and accessed by the agents. Section 5 then goes on to explain how this normative model can be incorporated into a BDI platform and how agents interact with the institutional model. We finish with related work (Section 6) and some conclusions and directions for future work (Section 7).

2 Case Study: Wireless Grids

The process and implications of modelling normative systems for agent reasoning can best be illustrated by a case study. The case study is set against the background of the next generation of mobile phones, where *wireless grids* have been proposed to address the energy issues inherent in these phones [7]. Batteries have fixed capacity that limits the operational time for a device. The increasing sophistication of mobile phones and their evolution into smart phones offering Internet access, imaging (still and video), audio and access to new services, has had a significant impact on power consumption, leading to shorter stand-by times, as well as the problem of rising battery temperature unless there is active cooling [19].

Wireless grids provide a mechanism that, in contrast to distributing digital content exclusively via an expensive (in terms of power and money) connection to a structured network, allows mobile phones to cooperate and share content via a cheap(er) ad-hoc connection[1] as well.

[1] In this paper we assume the use of the IEEE802.11 WLAN specification for the ad-hoc connection, which has the highest energy saving potential [19].

While providing energy gains, the scheme has the intrinsic weakness of distributed cooperative architectures: it depends on the cooperation of the participants to succeed. Cooperation in this context is understood as participants volunteering their resources, forming a common pool, that can be used by all to achieve a common goal, such as downloading a file. The utility that users can obtain from pooled resources is much higher than they can obtain on their own. Unfortunately this commitment comes at a cost, in the form of battery consumption for sending parts of files. As a consequence, (bounded) rational users would prefer to access the resources in the common pools without any commitment of their own, which puts the whole concept at risk.

Network users do not necessarily obediently cooperate by making their resources available without the prospect of rewards for good behaviour. Unreciprocated, there is no value to cooperation for a user. A lone cooperating user draws no benefit from its cooperation, even if the rest of the network does, as they receive parts via the cheaper connection. Guaranteed costs, paired with uncertainty or even lack of any resulting benefit does not induce cooperation in a (bounded) rational, utility-maximizing user. Without any incentives, rational users do not cooperate in such an environment and all will be worse off [1].

In this paper we show that an institution can be used to prototype and verify a cooperation mechanism—the design-time model—and subsequently use the mechanism to govern a simulation of a live system, using the run-time model. This two-phase approach demonstrates that we can build a norm-governed system that is: (i) *flexible:* by changing the institutional model, it is possible to influence agent behaviour, without modifying individuals—assuming a suitably goal-driven agent (ii) *realistic:* in this scenario, as in those foreseen for multi-agent systems, we cannot either predict or control with total certainty the behaviour of agents, but it is hoped that social institutions can provide functions similar to those found in the physical world, thus it is important to be able to test the potential impact of institutional control on suitably adapted agents.

3 Norm Governed Systems

Having briefly described the use case scenario, we now set out the formal model that we have adopted and its accompanying normative specification language.

3.1 The Institutional Model

We use the institutional model presented by Cliffe *et al* [3]. Its event-driven approach and mathematical foundation with computational support make it ideal for use in an agent-based simulation, as the actions of the agents advance both the simulation and the institutional state. The mathematical foundations provide verifiability and the underlying computational model using answer set programming allows for reasoning with incomplete information. In this section we provide an informal description of the model and its implementation.

The premise of the model is that events trigger the creation of institutional fluents. Inspired by Jones and Sergot's [15] account of institutional power and the notion of 'counts-as', the generation relation is used to define the connection between actions and their interpretation in the context of the institution. The effects of events, actions or institutional events – in terms of the initiation or termination of brute facts [14] and institutional fluents – is defined by the consequence relation. Thus, given an event and a state of the institutional model, represented as a set of (institutional) fluents, the next state can be determined by the composition of the transitive closure of the generation relation and the consequence relation.

The formal model is necessarily rather more detailed and precise than the sketch above. The essential elements of the institutional model are: (i) events ($\mathcal{E}$), that bring about changes in state, and (ii) fluents ($\mathcal{F}$), that characterise the state at a given instant, where a fluent is a term whose presence in the institutional state indicates it is true, and absence implies falsity. Fluents can either be initiated (i.e. become true) or terminated (i.e. become false) The model distinguishes two kinds of events: institutional events ($\mathcal{E}_{inst}$), that are the events defined by the model and exogenous ($\mathcal{E}_{ex}$), that are outside its scope, but whose occurrence triggers institutional events in a direct reflection of the counts-as principle. Institutional events are partitioned into institutional actions ($\mathcal{E}_{act}$) that denote changes in institutional state and violation events ($\mathcal{E}_{viol}$), that signal the occurrence of violations. Violations may arise either from explicit generation, from the occurrence of a non-permitted event, or from the failure to fulfil an obligation. The model also distinguishes two kinds of fluents: *institutional fluents* that denote institutional properties of the state such as permissions $\mathcal{P}$, powers $\mathcal{W}$ and obligations $\mathcal{O}$, and *domain fluents* $\mathcal{D}$ that correspond to properties specific to the institutional framework itself. The set of all fluents is denoted as $\mathcal{F}$.

The evolution of the state of the framework is achieved through the definition of two relations: (i) the generation relation, that specifies how the occurrence of one (exogenous or institutional) event generates another (institutional) event, subject to the empowerment of the actor. Formally, this can be expressed as $\mathcal{G} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{E}_{inst}}$, where $\mathcal{X} \subset 2^{\mathcal{F}}$ denotes a formula over the (institutional) state and $\mathcal{E}$ an event, whose confluence results in an institutional event, and (ii) the consequence relation, that specifies the initiation and termination of fluents subject to the performance of some action in a state matching some expression, or formally $\mathcal{C} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

The semantics is defined over a sequence, called a trace, of exogenous events. Starting from the initial state, each exogenous event causes a state change, through initiation and termination of fluents, that is achieved by a three-step process: (i) the transitive closure of $\mathcal{G}$ with respect to a given exogenous event determines all the (institutional) events that result (ii) to this, add all violations of events not permitted and all obligations not fulfilled, giving the set of all events whose consequences determine the new state, so that (iii) the application of $\mathcal{C}$ to this set of events, identifies all fluents to initiate and terminate with respect to

the current state in order to obtain the next state. So for each trace, a sequence of states is obtained constituting the model of the institutional framework.

Example 1. A very simple example could be the institution where pressing a button (exogenous event) indicates that you wish to buy (institutional event). Buying a good means you obtain ownership. Permission to buy is only obtained by registering (exogenous event). For this example we have that:

- $\mathcal{F} = \{\texttt{registered}, \texttt{own}, \texttt{pow(buy)}, \texttt{perm(buy)}, \texttt{perm(press)}, \texttt{perm(register)}\}$
- $\mathcal{E}_{ex} = \{\texttt{press}, \texttt{register}\}$
- $\mathcal{E}_{inst} = \{\texttt{buy}, \texttt{viol(buy)}, \texttt{viol(register)}, \texttt{viol(press)}\}$
- $\mathcal{G} = \{(\emptyset, \texttt{press}, \{\texttt{buy}\})\}$
- $\mathcal{C} = \{(\emptyset, \texttt{register}, (\{\texttt{registered}, \mathit{perm}(\mathit{buy})\}, \emptyset)), (\emptyset, \texttt{buy}, (\{\texttt{own}\}, \emptyset))\}$

Given an initial state $\Delta = \{\mathit{perm}(\mathit{press})\}$, the trace $\langle \texttt{register}, \texttt{press} \rangle$ does not lead to violations while the trace $\langle \texttt{press} \rangle$ produces `viol(buy)` indicating that perhaps the generate rules need to be adjusted such that buy only occurs if the user is registered.

3.2 Implementation

This formalization is realized as a computational model through Answer Set Programming (ASP) [11]. *AnsProlog*, the language of ASP, is a declarative knowledge representation language that allows the programmer to describe a problem and the requirements on the solutions. Answer set solvers like CLINGO [10] process the *AnsProlog* specification and return the solutions, in this case the traces, as answer sets. ASP permits, in contrast to related techniques like the event calculus, the specification of both problem and query as an executable program, thus eliminating the gap between specification and verification language. But perhaps more importantly, both – verification and specification language – are identical, allowing for more straightforward verification and validation. Cliffe *et al.* [3] show that the formal model of a normative framework can be translated to an *AnsProlog* program—a logic program under answer set semantics—such that the answer sets of the program correspond to the traces of the framework. A detailed description of the mapping can be found in [3].

Cliffe *et al.* [3] also put forward a domain-specific action language InstAL, for institution specification which compiles into *AnsProlog*. We use InstAL to describe our scenario. An InstAL program consists of two parts: the normative specification and a domain file. The normative specification consists of a static part and dynamic part. The former includes the name of the institution, and its events and fluents defined over a set of monomorphic types, such that it can be instantiated to create specific copies of the institutions for use by different (actual) participants. The dynamic part contains the description of the initial state of the institution which is instance specific. The range of values each type variable can have are specified in the domain file. The initial part of the specification identifies the fluents that comprise the normative state at the start.

InstAL uses semi-natural language to describe the various components of the normative framework. We introduce the language features of InstAL through fragments taken from the case study:

- `institution` *name* declares the name of the institutional framework, such as `institution grid`.
- `type` *identifier* declares a type, such as `type Handset`. Type declarations establish a disjoint set of monomorphic types. The types are specified in a domain file providing the acceptable values for each declared type. InstAL will then substitute those values whenever a type is specified for the events and fluents. An example is given in Fig. 5.
- `exogenous event` *event-name*(*type*$^+$) declares a new physical world event and the types of its parameters, such as `exogenous event download(Handset, Chunk, Channel)`.
- `inst event` *event-name*(*type*$^+$) declares a new institutional event and the types of its parameters, such as `inst event intDownload(Handset, Chunk, Channel)`.
- `violation event` *event-name*(*type*$^+$) declares a new violation event, such as `violation event misuse(Handset)`.
- `fluent` *fluent-name*(*type*$^+$) declares a new institutional fact—that is, an object that can be an element of the institutional framework state, such as `fluent downloadChunk(Handset, Chunk)`.
- `noninertial fluent` *fluent-name*(*type*$^+$) declares a fluent that is non-inertial, i.e. is not automatically persistent between states without termination. An example could be `noninertial fluent busyHSending(Handset)`.
- *event-name* `generates` *institutional-event*$^+$ [*condition*] adds a new pair to the generation relation with domain event (physical or institutional) and range institutional world event, subject to an optional condition. For example: `send (A, X) generates intSend(A) if hasChunk(A, X);`, where the condition is the presence of the fluent `hasChunk`, with the corresponding `A` and `X` (these variables are unified) in the institutional state.
- *event-name* `initiates` *institutional-fluent*$^+$ [*condition*] adds a new pair to the consequence (addition) relation, with domain event (physical or institutional) and range fluent. Thus, `instDownload(A, X, C) initiates hasChunk(A, X)` adds the corresponding fluent to the institutional state.
- *event-name* `terminates` *institutional-fluent*$^+$ [*condition*] adds a new pair to the consequence (deletion) relation, with domain event (physical or institutional) and range fluent. Thus, `intDownload(A, X, C) terminates pow(intDownload(A, X1, C1)` deletes the corresponding fluent from the institutional state.
- *non-inertial-fluent-name* `when` [*condition*] adds the conditions in which a non-inertial fluent should be true in a given state, like for example `busyHReceiving(A) when areceive(A, T2)`.
- `perm`(*event*) is a special fluent whose presence indicates that the event is permitted, such as `perm(intDownload(A, B, C))`, and is typically the subject of an `initiates` or `terminates` rule.

– `pow`(*event*) is a special fluent whose presence indicates that the event is empowered, such as `pow(intDownload(A, X1, C1))` (as above), and is typically the subject of an `initiates` or `terminates` rule.

InstAL also provides the notion of obligation, represented by the special fluent `obl(event, event, event)`., but since they are not used in the model under consideration, we do not discuss them further.

The declarative nature of ASP allows for a straightforward and efficient translation from the institutional requirements to a program that can compute all possible traces of the institution. Providing the same functionality using a procedural approach would be significantly harder.

4 Design-Time vs. Run-Time Models in the Wireless Grid Scenario

Most research to date on institutional modelling and reasoning focusses on the static properties of institutions. A model is used, for example, to determine whether a particular state of affairs is reachable or not from a given set of initial conditions. As such it can be used to design and verify properties of protocols and the effectiveness of sanctions. In our wireless grid scenario, the design-time model is used as a prototype to demonstrate that normative reasoning can be applied to the domain and to evaluate whether cooperation between the handsets is beneficial to the individual agents.

The design-time model is an abstraction of a possible running system and cannot take into account participants' reasoning capabilities as some of the participants might not be norm-aware or even irrational. In the design-time, it should be possible for participants to download the same chunk (i.e. part of the file) over and over again, while in reality this would be a waste of battery power. Additionally, the model does not have access to the information available in a running system so it might have to synthesize some information for itself. In the grid example, the base-station uses several different frequencies (frequency division multiplexing), and many users may download chunks simultaneously. We refer to a frequency division in the model as a channel. The design-time model has to keep track of which channels are in use at any given time in order to prevent simultaneous downloads on the channel. This also implies it has to monitor the duration of the download. The same is true for the sending and receiving of the chunks. In a running system, this is taken care of by the system and its components (such as the base-stations) or through the physical limitations of the devices.

The modelling of such extra details in the design-time model forces the designer to be very precise about his or her intentions, ultimately leading to better normative specifications.

While the design-time model assists the design of protocols, the run-time model assists in the running of and adherence to the protocol. The run-time model allows the participants to reflect upon the normative state of the system and their actions, enforces the normative specification by penalising violations

and helps in planning for agents' future actions taking into account normative repercussions.

For a given normative system, both the run-time and design-time model should have the same normative intentions, making the design-time model a good starting point for the development of the run-time one. A first step of the creation of the run-time model is to remove rules and conditions that deal with simulating a live system, because the purpose of the run-time model is to monitor the normative behaviour of participants, not the system's behaviour. It only monitors the external events resulting from agent actions, however, it does not predetermine all agent behaviour.

The design-time model might also contain enforcement mechanisms. In general, norm enforcement in a running system is the responsibility of the agents rather than the normative model. The only exception is the granting or removal of permissions to actions (the exogenous events) of the agents, which is part of the normative specification. In a running system, agent's actions can have side effects. Removing power from institutional events, limiting these in the design-time model might be sensible or useful in an design-time context but cannot be enforced in a running system. An example of this is removing the power of agents to receive chunks in the design-time model when they violate the sharing norm. In the running system this is impossible because the data is broadcast and asking an agent to penalise themselves is hardly sensible. A more appropriate sanction may be to exclude the agent from future interactions. While different in execution, both sanctions have the same result: the offending agent is shunned by the community—an approach which was previously demonstrated to be an effective enforcement mechanism [20].

To illustrate both the issues arising from design-time and run-time models, we describe each briefly in the following sections, highlighting aspects where they differ. Figs. 1-5 provide the full specification of both the design-time and the run-time model and an example of a domain file. As the run-time model is created from the design-time model by removing system data and enforcement mechanisms, we have chosen to show one specification, in which the run-time specification is in bold and the additional parts from the design-time model are in normal font; thus reinforcing that one is part of the other.

4.1 The Design-Time Model

The wireless grid scenario can be broken down into three phases (i) negotiation, where it is decided which handset needs to download which chunk from the base-station (ii) downloading, where a handset downloads a chunk from the base-station and (iii) sharing, where a handset broadcasts a chunk to receiving handsets. These three phases are distinct, but although negotiation must come first, obtaining and sharing can be interleaved as soon as downloading has commenced.

To avoid details that would unnecessarily complicate the specification, we impose the simplification that each file chunk is assigned to exactly one handset and that each handset is assigned the same number of chunks. Neither are

```
1  institution grid;
2
3  type Handset;
4  type Chunk;
5  type Time;
6  type Channel;
7
8  % exogenous event
9  exogenous event clock;
10 exogenous event download(Handset,Chunk,Channel);
11 exogenous event send(Handset,Chunk);
12
13 % creation event
14 create event creategrid;
15
16 % normative events
17 inst event intDownload(Handset,Chunk,Channel);
18 inst event intSend(Handset);
19 inst event intReceive(Handset,Chunk);
20 inst event transition;
21
22 % violation event
23 violation event misuse(Handset);
24
25 % fluents
26 fluent downloadChunk(Handset,Chunk);
27 fluent hasChunk(Handset,Chunk);
28 fluent areceive(Handset,Time);
29 fluent asend(Handset,Time);
30 fluent creceive(Handset,Time);
31 fluent csend(Handset,Time);
32 fluent transmit(Channel,Time);
33 fluent previous(Time,Time);
34
35 % Non-inertial fluents
36 noninertial fluent busyHSending(Handset);
37 noninertial fluent busyHReceiving(Handset);
38 noninertial fluent busyBReceiving(Handset);
39 noninertial fluent busyChannel(Channel);
```

Fig. 1. Declaration of types and events in the model

we concerned (for now) with the negotiation process that brings this about. A suitable allocation is given in the initial state of the model (see Fig. 4, lines 139–142) where the `downloadChunk` fluents indicate which handsets are tasked with downloading which chunks from the base-station. The handsets are also given the necessary permissions (lines 134–138). In the download phase each handset downloads its assigned chunks from the base-station.

The full specification of this phase is given in Fig. 2. Each handset can only obtain one chunk at a time from the base station, and each channel can only be used to download a single chunk. This is modelled using the non-inertial fluents `busyBReceiving` and `busyChannel` (lines 38–39) which are implied on the basis of the handset downloading and the base-station transmitting (lines 75–76). In a running system, this would be dealt with by the system rather than the normative model. The first InstAL rule (lines 47–49) indicates that a request to download a chunk is granted whenever there is an available channel, the handset is not currently receiving from the base-station and is not busy sending another chunk. When a chunk is downloaded, the handset and the channel are busy

```
41 %-----------------------------------------------
42 % rules for downloading
43 %-----------------------------------------------
44
45 % agent requests a chunk from the base station
46 % on channel C
47 download(A,X,C) generates intDownload(A,X,C)
48      if not busyChannel(C), not busyBReceiving(A),
49         not busyHSending(A);
50
51 download(A,X,C) generates transition;
52 clock generates transition;
53
54 intDownload(A,X,C) initiates hasChunk(A,X);
55
56 intDownload(A,X,C) initiates creceive(A,4),
57                              transmit(C,4);
58
59 transition initiates transmit(C,T2) if transmit(C,T1),
60                                        previous(T1,T2);
61 transition initiates creceive(A,T2) if creceive(A,T1),
62                                        previous(T1,T2);
63 transition initiates pow(intDownload(A,X,C))
64      if creceive(A,1);
65
66 intDownload(A,X,C) terminates pow(intDownload(A,X1,C1));
67 intDownload(A,X,C) terminates pow(intDownload(B,X1,C));
68 intDownload(A,X,C) terminates downloadChunk(A,X);
69 intDownload(A,X,C) terminates perm(download(A,X,C1));
70
71 transition terminates csend(A,Time);
72 transition terminates creceive(A,Time);
73 transition terminates transmit(C,Time);
74
75 busyChannel(C) when transmit(C,T2);
76 busyBReceiving(A) when creceive(A,T2);
```

Fig. 2. Generation and consequence relations for downloading

for a fixed amount of time—4 time steps in this case (lines 56–57). From the first instant of the handset interacting with the base-station, it is deemed to have downloaded the chunk, so parts can be shared (line 54). As soon as a channel and a handset are engaged, the institution (i) removes the power from the handset and from the channel to engage in any other interactions (lines 66–67), (ii) stops the handset from needing the chunk (line 68) and (iii) cancels the permission to download the chunk again later on (lines 69).

In the design-time case, we need a mechanism to mark the passing of time. For this purpose, each exogenous event generates a transition event (lines 51–52), while the `clock` event indicates that no handset was interacting with the institution. The `transition` event counts down the period of the interaction between the channel and handset (line 59–62). When the the interaction finishes, `transition` restores the power for a handset to download chunks via the channel and for the handset to download more chunks (lines 63–64). The event also terminates any busy fluents that are no longer needed (lines 71–73).

In the sharing phase each handset sends chunks to or receives chunks from another handset, with the goal that at the end of the process, each handset has a complete set of the chunks. The full specification is given in Fig. 3.

```
78  %-----------------------------------------------
79  % rules for sharing
80  %-----------------------------------------------
81
82  % Agent A broadcasts chunk X: This is broken down
83  % in sending and receiving internal actions
84  send(A,X) generates intSend(A) if hasChunk(A,X),
85          not busyHSending(A), not busyHReceiving(A),
86          not busyBReceiving(A);
87
88  send(A,X) generates intReceive(B,X)
89       if not hasChunk(B,X), not busyHSending(B),
90          not busyHReceiving(B), hasChunk(A,X),
91          not busyHSending(A), not busyHReceiving(A),
92          not busyBReceiving(A);
93
94  send(A,X) generates transition;
95  clock generates transition;
96
97  viol(intReceive(A,X)) generates misuse(A);
98
99  intReceive(A,X) initiates hasChunk(A,X);
100
101 intSend(B) initiates perm(intReceive(B,X));
102 intReceive(A,X) initiates areceive(A,2);
103 intSend(B) initiates asend(B,2);
104
105 transition initiates asend(A,T2)
106      if asend(A,T1), previous(T1,T2);
107 transition initiates areceive(A,T2)
108      if areceive(A,T1), previous(T1,T2);
109 transition initiates pow(intReceive(A,X))
110      if areceive(A,1);
111 transition initiates pow(intSend(A)) if asend(A,1);
112
113 intReceive(A,X) terminates perm(intReceive(A,X));
114 intReceive(A,X) terminates pow(intReceive(A,X));
115 intSend(A) terminates pow(intSend(A));
116
117 misuse(A) terminates pow(intReceive(A,X));
118
119 intReceive(A,X) terminates perm(intReceive(A,Y));
120
121 transition terminates asend(A,Time);
122 transition terminates areceive(A,Time);
123
124 busyHReceiving(A) when areceive(A,T2);
125 busyHSending(A) when asend(A,T2);
```

Fig. 3. Generation and consequence relations for sharing

The idea behind the model is similar to the downloading phase, but with two critical differences. First, the sending of one chunk by one handset automatically triggers the reception of the respective chunk by the partners (line 88), thus the design-time model assumes no network failures, etc. Furthermore, we build in a very basic mechanism to encourage handsets to share their chunks with others rather than just downloading them: when a chunk is received through sharing, the receiving handset loses permission to receive another chunk until it has sent a chunk (lines 119 and 101 respectively). Continuous receiving without sending (no permission is granted to `intReceiving`) results in a violation event named `misuse` (line 97). The simple penalty we chose to implement in our model is that

```
127 %-----------------------------------------------
128 % Initial state for downloading
129 % Dynamically generated in online case
130 %-----------------------------------------------
131 initially pow(transition), perm(transition),
132       perm(clock),
133       pow(intDownload(A,B,C)),
134       perm(intDownload(A,B,C)),
135       perm(download(alice,x1,C)),
136       perm(download(alice,x3,C)),
137       perm(download(bob,x2,C)),
138       perm(download(bob,x4,C)),
139       downloadChunk(alice,x1),
140       downloadChunk(alice,x3),
141       downloadChunk(bob,x2),
142       downloadChunk(bob,x4);
143
144 %-----------------------------------------------
145 % Initial state for sharing
146 % Dynamically generated in online case
147 %-----------------------------------------------
148 initially pow(transition), perm(transition),
149       perm(clock),
150       pow(intReceive(Handset,Chunk)),
151       pow(intSend(Handset)),
152       perm(send(Handset,Chunk)),
153       pow(intSend(Handset)),
154       perm(intReceive(Handset,Chunk)),
155       perm(intSend(Handset));
156
157 %-----------------------------------------------
158 % time
159 %-----------------------------------------------
160 initially previous(4,3);
161 initially previous(3,2);
162 initially previous(2,1);
```

Fig. 4. Initial state of the model, post negotiation

```
1 % This is dynamically created in the online case
2 Handset: alice bob
3 Chunk: x1 x2 x3 x4
4 Channel: c1 c2
5 Time: 1 2 3 4
```

Fig. 5. The domain information

the violating handset permanently loses the power to `intReceiving` (line 117), which means that for all intents and purposes it has been expelled from the group.

Fig. 5 shows the domain file for a scenario of two agents sharing four chunks. The traces generated by this design-time model verify that when agents follow the norms the entire community benefits—except if norms are breached at the end of the trace, as the penalty has no effect. While this might not cause problems if participants never meet again, penalties can always be applied at the next encounter. This information gives us sufficient reassurance to implement the protocol in our energy-saving simulation, where handsets might engage in several

sharing contracts over a period of time and past information can be used against them and propagated through the network.

4.2 The Run-Time Model

As mentioned earlier, for a given normative system, both the design-time and run-time model should have the same normative intentions. A first step for moving from the design-time model to the run-time one, is to remove rules and conditions that deal with simulating a live system. Thus, it only monitors the external events resulting from agents' actions.

As a consequence of moving to a run-time model, we no longer need to model system data. Concretely for our example, this means that the model does not have to track whether a channel is being used at a given moment or that a particular handset is incapable, from a technical perspective, of sending or receiving chunks. This means that the exogenous event `clock` and institutional event `transition` are no longer required (Fig. 1 lines 9 and 20). By the same reasoning, we longer need fluents to indicate that a handset or channel is engaged or to indicate elapsed time. This means that lines 28 to 39 in Fig. 1 are no longer necessary. With all of these events and fluents gone, the type `Time` is no longer needed.

In the design-time model, we penalize misbehaviour by taking away the power of a handset to receive chunks. While this may be a reasonable simplification in a design-time model for verification purposes, it cannot be enforced in a running system unless one expects agents to penalize themselves. Instead the system notes the violation and agents may use this information in future interactions with the offending agent. Thus, we remove the violation event `misuse` (Fig.1 line 23), its generate rule (Fig. 3 line 97) and any rules that terminate the power of agents.

In the run-time model, the assignment of chunks to agents (i.e. the initial configuration of the agent/chunk combinations indicated by the `initially` identifier in the InstAL specification) is determined at run-time by agents, which meet, decide to cooperate and negotiate which agent is to download and share which chunk. This information is then added to the fixed part (i.e. everything but the `initially` part of the InstAL specification).

4.3 Monitoring Run-Time State

For maintaining the normative state in our running system we introduce a special type of agent or entity: the InstitutionKeeper, which is conceptually similar to the well-know "Governor"-idea (see [17] for example). The task of the InstitutionKeeper is to monitor the actions of the agents in the context of the institution(s) in which they participate and consequently update the institutional state as a result of these actions. Agents can query this normative entity regarding the current state or possible future states of the institution, allowing them to take this information into account when deciding on their actions. In order to do so, the InstitutionKeeper is given the static part of the institution, which

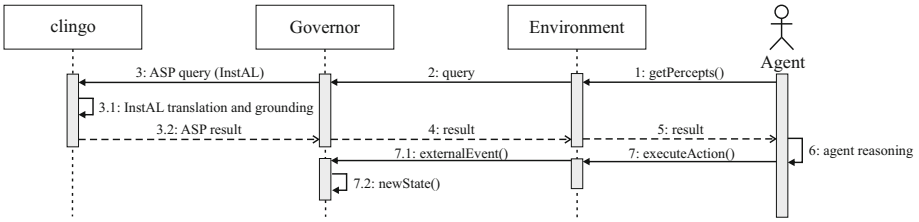


Fig. 6. Interaction between the various components

can then be instantiated when agents participating in it are known. This way a InstitutionKeeper can be responsible for as many instantiations of the same institutional specification as needed. We refer to these instantiations as contracts. In our wireless grid example, when two agents agree to collaborate they create a contract comprising their agent IDs, the chunks involved, the channels they can use and their allocation of which agent is downloading which chunks from the base station. This information is expressed as a custom domain file and an initial institutional state. Each contract is represented as a new instantiation of the institution. Whenever an action takes place that affects a contract, the InstitutionKeeper is informed of the agent ID and the action and computes the next normative state for that contract using the current state (for the `initially` part) and the associated domain file. Having the information for the initial contract, as well as tracking the normative state of each contract by analysing the respective exogenous events, the InstitutionKeeper can act as an institutional query processor for the agents. Contracting agents can query the current state and establish consequences of potential actions.

Currently we have three kinds of queries implemented that are useful for agents: (i) queries about the current state, including the norms applying to that state, (ii) queries about the possible impact of the agent's own actions, and (iii) general queries on what might happen in the future.

5 BDI Agents and Institutions

For the implementation of our simulation and the run-time reasoning of the agents we use Jason [2], a Java-based interpreter for an extended version of AgentSpeak. We link it to the institutional model and the answer set solver using system calls. The BDI architecture allows agents to reason easily about environmental and normative percepts alike while the Jason implementation takes care of the simulation side.

The UML sequence diagram in Fig. 6 shows the architecture of the run-time reasoning model and the dataflow in the simulation. The agents use the getPercepts() method to retrieve environmental percepts and normative percepts—the latter provided by the InstitutionKeeper. The InstitutionKeeper is implemented in Java and handles all the instantiations of the institution, storing

```
1  /* downloading */
2  +!check_download_channel: free(CHANNEL)
3  <- ?assigned_chunk(GN,X,N,CHUNKNUMBER,CHUNKSIZE);
4     check__permission_download(GN,CHUNKNUMBER,
5     CHANNEL);
6     !download_decision.
7
8  +!check_download_channel: true
9  <- .send(basestation, askOne, free(CHANNEL), Reply);
10    !check_download_channel.
11
12 +!download_decision: power(P) & permission(PE) &
13    P == true & PE == true
14 <- ?assigned_chunk(GN,X,N,CHUNKNUMBER,CHUNKSIZE);
15    ?event(E);
16    .send(basestation, tell, download(CHANNEL));
17    update_download_costs(CHUNKSIZE,E);
18    external_event_download(GN,CHUNKNUMBER,CHANNEL);
19    .send(basestation, untell, download(CHANNEL));
20    !sending_decision(GN,CHUNKSIZE,S,D).
21
22 +!download_decision: true
23 <- !check_download_channel.
24
25 /* sending */
26 +!sending_decision(GN,CHUNKSIZE,S,D):
27    assigned_chunk(_,_,N,_,_) & S <= D & N==3
28 <- ?assigned_chunk(GN,X,3,CHUNKNUMBER,CHUNKSIZE);
29    !check_violation(GN,CHUNKNUMBER);
30    !sending_decision_violation(GN).
31
32 +!check_violation(GN,CHUNKNUMBER): true
33 <- ?partner(PARTNER);
34    query_violation(PARTNER,CHUNKNUMBER,GN).
35
36 +!sending_decision_violation(GN): violation(V) &
37    V == true
38 <- check_missing_chunks.
39
40 +!sending_decision_violation(GN): true
41 <- ?assigned_chunk(GN,X,3,CHUNKNUMBER,CHUNKSIZE);
42    ?partner(PARTNER);
43    .send(PARTNER,tell,sharing(GN,CHUNKNUMBER));
44    update_sending_costs(CHUNKSIZE,GN,CHUNKNUMBER);
45    check_missing_chunks.
```

Fig. 7. Code fragment of the Agent Reasoning

the state of each. The state gets updated whenever an agent performs a relevant action by running the run-time model of the institution with the current state as initial state and obtaining the new state after the trace containing the action has been executed. Agent queries are answered by checking the current state for fluent presence or absence, or by executing the model of the institution with a complete or partial trace and examining the answer set(s) returned.

The code fragment in Fig. 7 shows how the agent can use normative information in its reasoning process. The central elements of Jason agent code are plans (i.e. steps an agent has to take in order to achieve its current intentions) that have the form $-/+$!intention: precondition $\leftarrow$ steps.

The first part of the agent code (lines 1–23) shows some of the agent's considerations for downloading a chunk. Notice that in contrast to the design-time

case, the agent is not automatically allocated a chunk, but has to check whether the base station has a free channel itself (lines 2–10). Beside this channel checking, the agent checks whether it has permission to perform the action (line 4) and being a norm-abiding agent, this agent only performs the action if it has both (line 11).

Lines 25–45 of the code fragment show the agent reasoning about sending its last (in our example the third) chunk. First, it checks whether receiving a chunk would be a violation for its partner (because the partner has not been sending) (line 29, 32–34). If so, the agent only checks which chunks it is missing and download these for itself (code not given). If the partner has been collaborating, the agent sends the agent the chunk (lines 43–44), if the costs are lower than a potential sanction for non-cooperation (precondition line 27).

6 Related Work

We contrast what we have presented here with several frameworks for organizational/institutional modelling and implementation, namely Opera/Operetta, $\mathcal{M}$OISE$^+$ Islander, OCeAN and the constraint framework of Garcia-Camino.

The focus in Opera [18] is the specification of (valid) normative states: how the states are achieved is not addressed, nor is the matter of which agents are responsible for the actions that brought about these states. The model on which our work is based ([3]) is complementary to Opera, focussing instead on actions and whether they are empowered or permitted and whether obligations are fulfilled, as well as maintaining a complete trace of the institutional history. Furthermore, the computational model put forward here enables a direct, model-checking approach, both to the verification of the design-time model and to its utilisation by agents enquiring about the institutional state as it evolves.

$\mathcal{M}$OISE$^+$ [13] is potentially the most similar to what we have set out: it is described as a middleware for organizational programming, which like us uses Jason as the agent framework and extends Jason to allow agents to perceive their organization. However, it would appear from [13], that agent autonomy is quite constrained by the roles they take on at any one time. In particular, it is notable that individual actions can be restricted by the group to which an agent belongs. In contrast, the agents in our model are simply subject to the norms of the institution and can choose whether to observe them or not.

Islander [6]—and associated tools—offer a comprehensive environment for the specification of electronic institutions through state diagrams labelled with speech acts, coupled with simulation via Ameli, supported by JADE. The agents are constructed from the specification and are in essence regimented by speech acts, rather than being autonomous, perceiving an illocution and then taking some action that may or may not have meaning in the (single) pervading institutional context. Although it would appear to be possible to author agents that can operate in this environment, using an internal architecture of choice, they would be without access to any formal model of the institution in which they act, and so are functionally blind. Garcia-Camino [9] adds a computational

model to this framework that is very similar to the one taken as a basis here, but the objective seems to be limited to institutional specification with no run-time connection in support of agent reasoning.

Fornara et al [8] describe OCeAN, which is a meta-model for institutions, describing the abstract syntax and semantics for the constructs that characterize an institution. The whole purpose of OCeAN is *not* to commit to a concrete language for specification, but instead to express the properties of a modelling language through UML and the Object Constraint Language. Similar to our approach described here, they distinguish between *natural* attributes (which we call *brute* facts) and *institutional* attributes (resp. fluents), while identifying an institutional state, changes in which are brought about by *actions* that are constrained by pre-conditions and effects. Thus there would appear to a fairly close correspondence between their (design-time) model and the concrete language used here, however it is only through the dynamics of a computational model, that agents can actually examine how the current state was achieved and evaluate possible futures.

The attraction of connecting BDI agents and norms goes back to at least [5], which discusses a form of deontic logic to accommodate norms and a pseudo-code extension of the BDI agent loop. However, practical connections appear to be few, and even these do not appear to offer actual implementations: (i) Meneguzzi and Luck [16] discuss how BDI agents may assimilate (changes in) permissions, prohibitions and obligations, but unlike here, do not appear to address the utilization of norms in the agent reasoning process (ii) Criado, Argente and Botti [4] also describe a form of norm assimilation process via bridge rules, so agents may recognize and acquire norms from their environment, but the recognition process is unclear, nor is it apparent whether the norms are acquired from observation or supplied by an institutional framework.

7 Discussion

In this paper we demonstrated how social institutions can be incorporated in a multi-agent simulation, and consequently, in live MAS, since the only difference between the two is the source of the events. To achieve this, the traditional design-time institutional model, used for verifying design-time properties of the system, can be reduced to a run-time model that just contains normative information and the relevant domain fluents. In a live system, one would expect one or several of the agents to police the community and enforce normative behaviour—just as in the physical world the law is enforced by judges and police officers—and to control the non-normative side of the system.

To use the run-time model in a live system, it needs to be encapsulated in a monitor object—which we call an InstitutionKeeper—whose sole purpose is to manage normative state and to answer queries from norm-aware agents.

In our simulation, one InstitutionKeeper object manages several instantiations of the same normative framework, which are referred to as contracts. We observe that often more than one normative framework is active within

an application. Furthermore, some of these may interact with one another. In [3], the authors present the concept of multi-institutions where events in one institution cause events in another or change another institution's state. Extension of the InstitutionKeeper to accommodate multi-institutional reasoning is an important part of future work, along with the issue of using conventional distributed systems techniques, such as replication, as a means to avoid the InstitutionKeeper becoming a bottleneck or single point of failure.

Acknowledgements. Tina Balke was partially supported by funding from the ePolicy Project, European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 288147.

References

1. Axelrod, R.: The emergence of cooperation among egoists. The American Political Science Review 75(2), 306–318 (1981)
2. Bordini, R.H., Wooldridge, M., Hübner, J.F.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology. John Wiley & Sons (2007)
3. Cliffe, O., De Vos, M., Padget, J.: Specifying and Reasoning About Multiple Institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006 Workshops. LNCS (LNAI), vol. 4386, pp. 67–85. Springer, Heidelberg (2007)
4. Criado, N., Argente, E., Botti, V.J.: A BDI architecture for normative decision making. In: van der Hoek, W., Kaminka, G.A., Lespérance, Y., Luck, M., Sen, S. (eds.) International Conference on Autonomous Agents and Multiagent Systems, pp. 1383–1384. International Foundation for Autonomous Agents and Multiagent Systems (2010)
5. Dignum, F., Morley, D.N., Sonenberg, L., Cavedon, L.: Towards socially sophisticated BDI agents. In: ICMAS, pp. 111–118. IEEE Computer Society (2000)
6. Esteva, M., de la Cruz, D., Sierra, C.: Islander: an electronic institutions editor. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 1045–1052. ACM (2002)
7. Fitzek, F.H.P., Katz, M.D.: Cellular controlled peer to peer communications: Overview and potentials. In: Cognitive Wireless Networks, pp. 31–59. Springer (2007)
8. Fornara, N., Viganò, F., Verdicchio, M., Colombetti, M.: Artificial institutions: a model of institutional reality for open multiagent systems. Artif. Intell. Law 16(1), 89–105 (2008)
9. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.W.: Constraint rule-based programming of norms for electronic institutions. Autonomous Agents and Multi-Agent Systems 18(1), 186–217 (2009)
10. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-Driven Answer Set Solving. In: Proceedings of IJCAI 2007, pp. 386–392 (2007)
11. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3-4), 365–386 (1991)
12. Grossi, D., Aldewereld, H., Dignum, F.: *Ubi Lex, Ibi Poena*: Designing Norm Enforcement in E-Institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006 Workshops. LNCS (LNAI), vol. 4386, pp. 101–114. Springer, Heidelberg (2007)

13. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE. IJAOSE 1(3/4), 370–395 (2007)
14. Searle, J.R.: The Construction of Social Reality. Allen Lane, The Penguin Press (1995)
15. Jones, A.J., Sergot, M.: A Formal Characterisation of Institutionalised Power. ACM Computing Surveys 28(4), 121 (1996)
16. Meneguzzi, F.R., Luck, M.: Norm-based behaviour modification in BDI agents. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) International Conference on Autonomous Agents and Multiagent Systems (1), pp. 177–184. International Foundation for Autonomous Agents and Multiagent Systems (2009)
17. Noriega, P.: Agent mediated auctions: The Fishmarket Metaphor. Ph.D. thesis, Universitat Autònoma de Barcelona (1997), http://www.lania.mx/~pablo/articles/tesis/thesis.pdf
18. Okouya, D., Dignum, V.: Operetta: a prototype tool for the design, analysis and development of multi-agent organizations. In: International Conference on Autonomous Agents and Multiagent Systems (Demos), pp. 1677–1678. International Foundation for Autonomous Agents and Multiagent Systems (2008)
19. Perrucci, G.P., Fitzek, F.H., Petersen, M.V.: Energy saving aspects for mobile device exploiting heterogeneous wireless networks. In: Hossain, E. (ed.) Heterogeneous Wireless Access Networks: Architectures and Protocols, pp. 277–303. Springer US (2009)
20. de Pinninck, A. P., Sierra, C., Schorlemmer, W.M.: Distributed Norm Enforcement: Ostracism in Open Multi-Agent Systems. In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.) Computable Models of the Law. LNCS (LNAI), vol. 4884, pp. 275–290. Springer, Heidelberg (2008)

Modelling and Monitoring Interdependent Expectations

Stephen Cranefield[1], Michael Winikoff[1], and Wamberto Vasconcelos[2]

[1] Department of Information Science, University of Otago, Dunedin 9054, New Zealand
{scranefield,mwinikoff}@infoscience.otago.ac.nz
[2] Department of Computing Science, University of Aberdeen, AB24 3UE, Aberdeen, UK
wvasconcelos@acm.org

Abstract. Previous research on modelling and monitoring norms, contracts and commitments has studied the semantics of concepts such as obligation, permission, prohibition and commitment; languages for expressing behavioural constraints (such as norms or contracts) to be followed by agents in specific contexts; and mechanisms for run-time monitoring of fulfilment and violation of these constraints. However, there has been little work that provided all of these features while also allowing the current expectations of agents and the fulfilment and violation of these expectations to be expressed as first-class constructs in the language. This paper demonstrates the benefits of providing this capability by considering a variety of use cases and demonstrating how these can be addressed as applications of a previously defined temporal logic of expectations and an associated monitoring technique.

1 Introduction

Much research in multi-agent systems has been influenced by organisational principles from human society, and in particular social concepts such as norms and commitments have been extensively studied due to their potential to enable the efficient specification and management of agent interaction in open societies of autonomous agents.

Previous research on modelling and monitoring norms, contracts and commitments has studied the semantics of concepts such as obligation, permission, prohibition and commitment; languages for expressing behavioural constraints (such as norms or contracts) to be followed by agents in specific contexts; and mechanisms for run-time monitoring of fulfilment and violation of these constraints. However, there has been little work that provides all of these features while also allowing the existence, fulfilment and violation of obligations and commitments to be expressed as first-class constructs in the language. We believe that the ability to directly express statements about these features of an agent's social context is important as it allows the investigation of richer types of norms and contracts that are interdependent. Our aim in this paper is to demonstrate that this is a capability that is desirable but not adequately addressed to date, and to show how a logic and monitoring technique developed in our previous work can meet our requirements.

In this paper, we are not concerned with distinctions between norms and commitments, and generalise both concepts to the notion of *expectations* on future world states, events and/or agent actions, while ignoring social issues such as where these expectations come from (e.g. mandated by authorities or requested and accepted via agent

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 149–166, 2012.

messaging) and how they are embedded in the relationships that exist between agents. In our view these issues can be largely decoupled from the questions of what it means to have an expectation that is active, fulfilled or violated, and how these expectations change from one state to the next.

Expectations can also exist for reasons other than agents creating commitments or being subject to norms. A team tactic in a sport involves each involved player playing its role under the expectation that each other participant will correctly play their role. If a player detects the violation of this expectation (e.g. if a player falls over), then an alternative tactic must be initiated. An agent may also choose to base its practical reasoning based on expectations inferred through observation and experience, but may wish to monitor the status of these expectations. The semantics and uses of expectations have been studied by a number of researchers [14,3,18,32,41]. In this paper, the focus is on the representation and semantics of expectations and their fulfilment and violation.

The structure of this paper is as follows. In Section 2 we present a survey of a range of approaches to modelling and monitoring various types of expectations. Section 3 provides an overview of our previously defined logic of expectations, for which we have built an associated model checker for monitoring expectations, and explains why previously imposed restrictions on the nesting of expectation-related modalities can be lifted. Some use cases illustrating the utility of modelling interdependent expectations are presented in Section 4. Section 5 demonstrates that the expectation monitor we have developed in previous work, after a simple extension, can handle nested expectations, and Section 6 concludes the paper.

2 Previous Work

A wide variety of approaches have been investigated for modelling and monitoring constraints on agents' future behaviour in the context of electronic institutions, normative multi-agent systems and commitment-based semantics for agent communication. Early work on electronic institutions [23,38] focused on the development of middleware that can directly interpret an institution specification provided by a designer and ensure that agents follow the norms, and for this reason considered norm representations that have a procedural rather than declarative flavour, giving rise to the so-called "protocol-based norms". Work in the related field of normative multi-agent systems [11] has tended to focus on higher-level declarative representations of norms. Research on commitment-based semantics for agent communication [35,42] aims to explain the individual speech acts and/or complete dialogs exchanged between agents in terms of the commitments requested and made by one agent towards another.

There are strong links between these research fields with much work crossing the boundaries between them, e.g. the design of a norm representation language that includes operational details such as violation checks and repair strategies alongside a declarative norm [39], the extension of e-institution middleware to handle rule-based norms as well as protocol-based norms [27], and an institution specification language that models both norms and agent communications in terms of commitments [25].

Below we discuss work in these areas, focusing on the formalisms used, whether concepts such as expectation, fulfilment and violation are expressible in those formalisms, and whether (and how) the monitoring of expectations has been addressed.

The approaches discussed below range from high-level logical models, investigated mainly to gain semantic understanding of norms, commitments or general expectations, to operational models that can be directly executed and are therefore amenable to runtime monitoring. However, there has been no work that provides a good semantic account of the activation, fulfilment and violation of expectations of any sort, allowing these concepts to be explicitly represented, and also providing a technique for monitoring expectations, except for the work of Governatori and Rotolo [29], which addresses only the recovery from violations via contrary-to-duty norms. In Section 3 we show how our prior work on modelling and monitoring expectations can be extended to provide all three features, and argue why this ability opens up a new range of interesting use cases in expectation modelling and monitoring.

2.1 Logical Approaches

Shoham and Tennenholtz [34] addressed the problem of how to specify the behaviour of agents acting concurrently (and, in their framework, synchronously). Each agent is modelled by a set of possible states, a set of actions it can perform, and a transition function that maps a state, an action, and a set of *social laws* to a set of possible next states. A social law is a set of constraints associated with actions, where each constraint is a first order logical formula specifying the conditions (in terms of states) under which the agent is prohibited from performing the action. The aim was to enable the coordination of the agents' actions without the need to specify global system transitions. They studied a particular problem in the design of social laws. Later work has adapted this concept to multi-agent systems modelled by Kripke structures, e.g. to detemine the influence that each agent has in the success of a given *normative system* leading to a desired system objective (specified in CTL) [1]. In this later work, the normative system is specified semantically—as a subset of the transition relation representing forbidden transitions—rather than syntactically.

Dignum and colleagues investigated the extension of dynamic [21] and temporal [12] logics with deontic concepts to allow the expression of obligations involving deadlines. The obligations studied address either the performance of specific actions [21] or the fulfilment of (atemporal) propositions [12] by a deadline. In one approach [21], the semantics of formulae were defined relative to a state and a trace so that "the history (i.e. the trace) of an ideal world might differ from the history of the present world", and this feature was used to define the notion of ideality represented by obligations. Later work used simpler semantics in which models of the logic are assumed to include a propositional constant $Viol$[1]. Broersen et al. [12] also used an ideality proposition Idl to allow a more subtle account of deadline obligations. These propositions have no semantics of their own—they are given semantics indirectly via the definitions of the obligation operators, which constrain the states in which $Viol$ and Idl should hold.

[1] In some work it is noted that this propositional constant could be qualified, e.g. with a norm index [20], so that different types of violation can be distinguished.

The research cited above did not address the modelling of obligations dependent on the fulfilment or violation of other obligations, except for the simple case of the violation or fulfilment of single actions. Also, in the examples of interdependent obligations considered in this [21] and earlier work [20], rather than explicitly using $Viol$ and Idl predicates as conditions of norms, predicates directly expressing the occurrence or lack of occurrence of the specific desired action are used. While this demonstrates how, for specific examples, an obligation can be made conditional on a predicate that happens to correspond to fufilment or violation of another obligation, there is no systematic treatment of nested violation and fulfilment operators within obligations. This is reasonable given the restricted setting (obligations to perform a given action), but this approach leaves open the question of how inter-related norms with more complex temporal structure could be expressed.

Raimondi and Lomuscio [33] present an approach for model checking multi-agent system specifications written in a logic combining epistemic, deontic and temporal operators, with semantics based on the interpreted systems model enhanced by partitioning the agents' local states into two disjoint sets of allowed and disallowed states (which are specified in an MAS definition). Two operators are interpreted in terms of the allowed states, a deontic operator $O_i\phi$ (under all the correct alternatives for agent i, ϕ holds) and a knowledge operator $\widehat{K}_i^j\phi$ (agent i knows ϕ under the assumption that agent j is functioning correctly). A standard epistemic operator is also included. The aim is to verify statically specified properties of specifications rather than to represent and monitor run-time behaviour.

Alberti et al. [3] describe a means to perform run-time protocol compliance monitoring based on logical constraints expressing positive and negative expectations as the consequences of observed actions. At run time, agent messages are detected and asserted as facts, and abductive inference [2] is used to keep track of pending, fulfilled and violated expectations. However, this information about the state of expectations cannot be expressed using constraints, so interdependent expectations cannot be modelled.

Verdicchio and Colombetti [40] use a variant of CTL* to provide axioms defining the lifecycle of commitments that arise through the exchange of messages. The language includes predicates to represent a commitment being made and whether it is fulfilled, violated or pending. It seems that these predicates could appear within the content of commitments. There is no discussion of how the language could be used for practical reasoning.

Bentahar et al. [8] define model-theoretic semantics for their Commitment and Argument Network (CAN) formalism that models agent communication in terms of social commitments and argumentation. Their logical language can express the creation of commitments of various sorts and requests for commitments to be made, as well as the satisfaction and violation of commitments. It appears that the satisfaction and violation operators can be nested within the content of a commitment. Their discussion of pragmatic aspects of their formalism [7] does not address the monitoring of commitments.

Singh [35] provides model-theoretic semantics for commitments, with two modalities defining practical and dialectical commitments between a debtor x and creditor y. The language allows these modalities to be nested. The paper discusses possible reasoning postulates and their soundness and completeness, but there is no discussion of

how this logic could be used in practice. It is, however, claimed that the approach provides a basis for specifying precisely how commitments arise in a context and can be manipulated. Modalities corresponding to fulfilment and violation are not discussed.

Governatori and Rotolo [29] propose a technique for design-time checking of a set of rules specifying some process against a set of normative rules regulating it. This is in contrast with the run-time checking of actual behaviour that is the focus of this paper, and is therefore contingent on the process descriptions to be checked being available for this purpose. The normative language used is based on defeasible logic and has a special focus on "contrary to duty" norms. It thus has an implicit notion of violation of a norm expressible in the language.

Dastani et al. [19] describe an assertion-based debugging technique for BDI-based multi-agent programs. They use an assertion language based on linear temporal logic (LTL) that can refer to agents' beliefs, goals, plans and perceived events. In this approach, assertions are added to individual agent programs, and are associated with a particular debugging tool (such as a breakpoint or watch tool). An extended agent interpreter evaluates the assertions against the current trace of the multi-agent system. As this is a debugging technique designed to provide information to human designers rather than agents, there is no explicit notion of fulfilment or violation within the assertion language. Also, social concepts such as norms and commitments are not addressed.

Cranefield and Winikoff [18] define an extension of propositional linear temporal logic that includes temporal operators stating that an expectation currently exists, is fulfilled, or is violated as a result of a particular conditional rule of expectation. In contrast to the work discussed above, the concepts of violation and fulfilment of expectations are given their own first class semantics. The logic, as described previously, did not allow formulae representing existence, fulfilment or violation of an expectation to appear nested within a rule of expectation; for example, a rule could not be triggered by the violation of another rule. A model checking procedure allows the truth of these formulae to be determined either off-line (e.g., when checking an audit trail) or incrementally as new states become available. This logic is the basis of the discussion in this paper, and a modified version is described in Section 3.

2.2 Rule Languages

García-Camino et al. [27] present a language for defining conditional norms and the sanctions or rewards associated when norms are fulfilled or violated. Norms control the utterance of speech acts within particular periods (specified in terms of dates or relative to other speech acts). Sanctions can modify attributes of an agent, such as its available credit. The language is given an operational semantics in terms of the Jess expert system shell, and this allows norm fulfilments and violations to be detected at run-time and sanctions to be applied. However, the occurrence of fulfilments and violations cannot be expressed within the normative rules themselves.

Boella and van der Torre [10] propose a normative systems approach for defining document access control policies in distributed virtual communities. Community and individual policies are represented using beliefs, desires and goals described using rules of the form $l_1 \wedge \cdots \wedge l_n \rightarrow l$, where l and each l_i are literals built from a set of

propositional decision variables and system parameters. Decision variables may be defined to represent the situation where a given agent is considered to be in a state of violation if some specific variable is *true*, and this allows nested violations to be represented. For example, agent a_2 may be deemed to be in violation if it does not consider the state of affairs represented by variable q to be a violation by agent a_1. There is no underlying semantics for violation—the determination that a violation exists is local to an agent based on its rules. This approach allows the negotation of agents for access to information to be characterised using game-theoretic techniques.

García-Camino et al. [28] define an expressive rule language with constraints for specifying conditional norms and explicitly tracking the normative state of a multi-agent system as agents exchange messages. Rules may refer to norms, so it is possible to define rules stating, for example, that one obligation triggers another. Although the rules track the normative state of the multi-agent system and therefore detect violations of norms, these cannot be represented using the proposed set of predicates for representing normative states. As the rule language is not dependent on the predicates used to model states, additional fulfilment and violation predicates could be added. However, the only semantics for predicates are any operational ones defined by rules.

Fornara et al. [25] describe an approach for specifying institutions in which agents communicate. The content of commitments comprise an action, proposition or referential expression existentially or universally bound to an interval of time. Norms are event-driven rules to create, update or cancel commitments. Although the authors advocate the suitability of an operational approach to checking agent norm-compliance, they do not give details as to how this could be done, and their language cannot express fulfilments and violations of commitments and norms.

Tinnemeier et al. [37] propose an approach for specifying normative artifacts: coordination artifacts that are endowed with a knowledge of a set of norms and the ability to impose sanctions. The specification defines actions in terms of pre- and post-conditions expressed in terms of the brute (non-normative) state of the system. In addition, there is a set of norm schemes with parameterised labels that are conditional on either the brute state or the violation of another norm (using its label)—this latter feature allows contrary-to-duty situations to be specified. The body of a norm scheme specifies an obligation or a prohibition, and the scheme also includes a deadline and a sanction, which are represented by conjunctions of literals. The sanction represents a change that will be made to the brute state if the norm is violated. The execution of normative artifacts is specified using operational semantics.

Aldewereld et al. [4] have considered the use of "counts-as" predicates to link normative (abstract) events with real-world (concrete) events. In particular, obligation and prohibition norms in deontic logic are operationalised as "counts-as" statements: an obligation (respectively prohibition) with content ϕ maps to the statement that $\neg\phi$ (respectively ϕ) counts as a norm violation. Norms can have a maintenance condition (once active, the norm is deemed violated if this condition evaluates to false), and this allows a limited degree of temporal expressiveness. It is not clear if the violations of different norms can be distinguished and there is no discussion of whether the norm violation and fulfilment conditions can be used within the content of norms. The approach is implemented using the DROOLS forward-chaining engine.

2.3 Action Description Languages

Yolum and Singh [43] use the event calculus to define the social semantics of agent interaction protocols in terms of commitments. An event calculus planner is then used to generate possible execution paths of the protocol in which all pending commitments have been resolved. Commitments are modelled as atemporal propositions with a debtor and creditor, and there is no explicit representation of violation.

Chesani et al. [15] also provide commitment-based semantics for agent interaction protocols using the event calculus, but focus on run-time monitoring of commitments. To allow this, they implement the cached event calculus [16] using the abductive proof procedure $\mathcal{S}$CIFF [2], which includes positive and negative expectation predicates. Commitments are expressed using the approach of Yolum and Singh extended with temporal constraints that allow deadlines to be expressed.

Artikis and Sergot [5] use the event calculus for specifying and tracking normative states of multi-agents systems based on the concepts of obligation, power and permission. Their approach specifies how the actions agents perform affect the values of fluents (dynamic properties) encoding the state of the domain and the powers, permission and obligations of the actors. Obligations represent actions that agents should perform (rather than states of the world they should bring about). A violation fluent is used to declare that an action causes a violation, but this has no special semantics. Farrell et al. [24] present a similar approach for modelling and monitoring the state of contracts.

Cliffe et al. [17] present an approach for modelling norms using an action description formalism based on events, fluents, causal rules (similar to those in the event calculus) and generation rules (which describe when certain events count as other, institutional, events). Fluents representing institutional power, permission and obligation are used to define the effects of events in the application domain. The semantics of violation are given by generic rules that define how violation events are triggered when actions are not permitted or obligations aren't satisfied by their deadline. There is a violation event defined for each normative action and each non-normative event, and it appears that these events could be included in the conditions of rules. Answer set programming is used to perform queries about possible traces of a specified normative framework that satisfy certain properties of interest.

Commitment machines [42] define agent interaction protocols by specifying the preconditions and effects of the agent actions in terms of commitments that exist between participants. A set of protocol states are defined in terms of the propositions and commitments that hold in them and domain actions are defined in terms of the propositions and commitments they cause to hold (their "effects"). Actions cause transitions between states if the new state is a logical consequence of the original state and the action's effects. Agents interpret commitment machines at run time to determine a desired path through the protocol, or they can execute a finite state machine compiled from the commitment machine. There is no notion of violation of a commitment in this formalism—an execution of the protocol either reaches a state in which a desired commitment exists, or it does not. Commitments can be conditional on the existence of other commitments, but these represent instances of conditional commitments between agents that were created during the protocol execution, not general rules that one commitment should always create another.

2.4 Automata-Based Approaches

Spoletini and Verdicchio [36] present an automata-based technique for monitoring commitments expressed in a propositional temporal logic with both past and future operators. The monitoring problem is modelled as a word recognition problem over an alphabet comprising propositions representing the contents of "sniffed" agent messages and the values of past-oriented subformulae of the formula to be monitored. The formula is preprocessed using Gabbay's rules [26] to separate out any future operators nested within past operators. The values of subformulae formed from past operators with no nested future operators are recognised dynamically by deterministic Büchi automata, and these subformulae are replaced by special propositions representing the outputs of the automata. The resulting formula is then translated into an alternating modulo counting automata. In this approach, fulfilment and violation are represented by the operational condition of the automaton reaching an acceptance or non-acceptance state—there is no representation of fulfilment or violation within the language used for representing commitments.

Lomuscio et al. [30] describe an approach to modelling contracts governing Web services using timed automata with discrete data (TADD). These automata include a partition of the states into ideal and non-ideal ones and a set of integer variables. Run-time monitoring of contracts is done by collecting system snapshots comprising valuations for the variables and checking these against the TADD encoding the contract to determine whether each execution step is in conformance with the contract. Like the work of Spoletini and Verdicchio, this symbolic approach avoids the need to keep an execution history in memory. However, the trade-off is that there is no declarative representation of the contract and its current state for use in an agent's reasoning.

Modgil et al. [31] model norms with augmented transition networks (ATNs), comprising three states representing the norm being inactive, active and either fulfilled or violated. ATNs are processed via an architecture in which observer agents send messages to monitors, which trigger transitions in the ATNs and notify a manager agent of norm fulfilments and violations. The norms could, in principle, include messages announcing fulfilments and violations in arc labels, with the manager having the responsibility of sending these, but this extension is not proposed in the paper. The approach is defined in terms of a highly procedural account of the architecture and the interaction between its components, and it is difficult to relate it to more declarative approaches.

3 A Temporal Logic of Expectation

The logic used in this paper is based on an extension of propositional linear temporal logic proposed by Cranefield and Winikoff [18]. However, in this paper we introduce some changes[2] from the original presentation and omit some features of the language

[2] The syntax of the previous version of the logic did not include four-argument versions of Exp, Fulf, Viol nor Trunc_S and Progress operators. However, these operators were defined semantically and used in the definitions of the two-argument versions of Exp, Fulf and Viol (which we have renamed here from their original names ExistsExp, ExistsFulf and ExistsViol). Here, to allow a concise presentation, we include all these operators in the syntax.

that are not relevant to the discussion in this paper. The syntax of the logic is described by the following grammar:

$$\begin{aligned}\phi ::= {} & \mathsf{Exp}(\phi,\phi,n,\phi) \mid \mathsf{Fulf}(\phi,\phi,n,\phi) \mid \mathsf{Viol}(\phi,\phi,n,\phi) \mid \\ & \mathsf{Exp}(\phi,\phi) \mid \mathsf{Fulf}(\phi,\phi) \mid \mathsf{Viol}(\phi,\phi) \mid \\ & p \mid \neg\phi \mid \phi \wedge \phi \mid \bigcirc\phi \mid \ominus\phi \mid \phi \,\mathsf{U}\, \phi; \mid \phi \,\mathsf{S}\, \phi \mid n \mid \mathsf{Trunc}_{\mathsf{S}} \mid \mathsf{Progress}(\phi,\phi)\end{aligned}$$

where p is a proposition, $\bigcirc$ is the standard temporal "next" operator, $\ominus$ is the standard temporal "previous" operator, U is the standard temporal "until" operator, S ("since") is a backwards-looking version of until, and n is a nominal: a proposition that is constrained to be true in exactly one state in the model. We assume that the model contains at least one nominal for each state, as these are used in the semantics to identify the states in which "rules of expectation" fire and introduce new expectations. Nominals are a feature of hybrid logic [9], and the original version of the logic [18] contained other hybrid logic constructs. However, only nominals are needed in this paper. The $\mathsf{Trunc}_{\mathsf{S}}$ and $\mathsf{Progress}$ operators are explained below.

We assume the propositions include $\top$ (true) and $\bot$ (false), with their usual meanings, and define as abbreviations the Boolean connectives $\vee$ and $\rightarrow$, the derived temporal operators "eventually ϕ" ($\Diamond\phi \equiv \top \,\mathsf{U}\, \phi$), and "always ϕ" ($\Box\phi \equiv \neg\Diamond\neg\phi$), and similar backwards-looking versions ($\blacklozenge\phi \equiv \top \,\mathsf{S}\, \phi$ and $\boxminus\phi \equiv \neg\blacklozenge\neg\phi$).

The semantics determine the truth of a formulae at a given state in a model comprising a finite or infinite sequence of states together with a valuation function specifying the propositions that hold in each state. In the case of a finite model, either *strong* or *weak* semantics can be used to evaluate the $\bigcirc$ and U operators [22]. The strong semantics assume a formula is false if the model does not include enough states to evaluate a formula, while the weak semantics assume a formula is true in this situation. Thus, in the final state of a finite model, $\bigcirc p$ is false under the strong semantics and true under the weak semantics. The operator $\mathsf{Trunc}_{\mathsf{S}}$ is a simplified form of an operator defined by Eisner et al. [22], and its semantics truncate the model at the current state and use the strong semantics to evaluate its argument formula. Essentially this means to determine whether the argument formula can be known to be true without using any information in future states. Formally, $\mathsf{Trunc}_{\mathsf{S}}\,\phi$ is true in state i of a model $\mathcal{M}$ if and only if ϕ is strongly true ($\models^{+}$) in a truncated model $\mathcal{M}^i$ where all states after i have been removed:

$$\mathcal{M}, i \models \mathsf{Trunc}_{\mathsf{S}}\,\phi \quad \text{iff} \quad \mathcal{M}^i, i \models^{+} \phi$$

3.1 Expectation Operators

The first two arguments, of the Exp, Fulf and Viol operators represent a conditional rule of expectation. Although the condition and expectation of a rule always appear as separate arguments of an operator in our logic, for convenience we will write $\lambda \Rrightarrow \rho$ as shorthand[3] for "the rule given by the pair λ and ρ". The meaning of a rule $\lambda \Rrightarrow \rho$ is that if λ evaluates to *true* in any state, given the information in the model up to that state, then ρ is an expected constraint on the model at that state.

[3] Note that '$\Rrightarrow$' does not represent logical implication and is not formally part of our language.

Unlike most approaches to modelling norms and commitments, our expectations are not limited to propositions that describe a desired property of a single state (e.g. the performance of a given action by an agent) in conjunction with a simple deadline constraint. Instead we aim to study the fulfilment and violation of more general types of expectation, such as those that aren't brought about by agents' actions ("The sun will rise each morning") and those with complex temporal structure ("If I pay for a subscription then the publisher will send me a magazine issue each month for a year from the month after my payment is received"). Thus, λ and ρ can be any formula in our logic, although the semantics ensure that the rule can only fire if the condition λ can be evaluated without the use of information from future states[4]. The expectation ρ can be a formula expressing desired properties of the states up to the present and/or a constraint on the future sequence of states that should be monitored for fulfilment or violation.

A formula $\mathsf{Exp}(\lambda, \rho, n, \phi)$ means that the formula ϕ is an active expectation as a result of the rule $\lambda \Rrightarrow \rho$ having fired (i.e. its condition λ becoming true) in a (possibly prior) state specified by nominal n. If the rule fired in a prior state but the expectation was not immediately fulfilled or violated, then the current form of the expectation ϕ may be different from the expectation ρ in the rule due to the use of *formula progression* (explained below) to carry forward an expectation from one state to the next.

The operators $\mathsf{Fulf}(\lambda, \rho, n, \phi)$ and $\mathsf{Viol}(\lambda, \rho, n, \phi)$ have the same argument structure as Exp, and mean that the rule $\lambda \Rrightarrow \rho$ firing in the state specified by n has resulted in an active expectation ϕ that is fulfilled or (respectively) violated in the current state. These three operators are defined as follows (where n is a nominal):

$$\begin{aligned}
\mathsf{Exp}(\lambda, \rho, n, \phi) \iff & (n \wedge \mathsf{Trunc}_{\mathsf{S}}\ \lambda \wedge \phi \equiv \rho) \vee \\
& \exists \psi \ominus (\mathsf{Exp}(\lambda, \rho, n, \psi) \wedge \\
& \quad \neg \mathsf{Trunc}_{\mathsf{S}}\ \psi \wedge \neg \mathsf{Trunc}_{\mathsf{S}}\ \neg \psi \wedge \mathsf{Progress}(\psi, \phi)) \\
\mathsf{Fulf}(\lambda, \rho, n, \phi) \iff & \mathsf{Exp}(\lambda, \rho, n, \phi) \wedge \mathsf{Trunc}_{\mathsf{S}}\ \phi \\
\mathsf{Viol}(\lambda, \rho, n, \phi) \iff & \mathsf{Exp}(\lambda, \rho, n, \phi) \wedge \mathsf{Trunc}_{\mathsf{S}}\ \neg \phi
\end{aligned}$$

The definition of Exp states that there are two ways for an expectation to result from a rule $\lambda \Rrightarrow \rho$: either λ holds in the current state (without recourse to future information) and therefore ρ is now expected (i.e. it is an expected constraint on the model), or some other formula ψ was expected in the previous state as a result of the rule, ψ was not known to be true or false in that state given the model up to that point, and thus a "progressed" form of ψ is now expected. Progress is a temporal operator corresponding to the progression function defined by Bacchus and Kabanza [6] for planning with "temporally extended goals". Details are beyond the scope of this paper, but essentially, progression transforms a temporal formula from the viewpoint of one state into the viewpoint of the next state. A formula that can be determined to be true (respectively false) without recourse to any future states progresses to $\top$ (respectively $\bot$). A formula that requires future information in order to be fully evaluated is partially evaluated using information from the model up to the current state and is then re-expressed

[4] Future states might be available in offline monitoring of expectations, such as the examination of an audit trail.

as an equivalent constraint in the context of the next state. For example, if p holds in the current state, then $p \wedge \bigcirc q$ progresses to q, expressed as $\mathsf{Progress}(p \wedge \bigcirc q, q)$.

The Exp, Fulf and Viol operators defined above are rather specific in the information they express about a currently active, fulfilled or violated expectation: the third and fourth arguments record the state in which the rule's condition became true and the current form of the expectation. In many cases, it may be sufficient to know there is currently an active, fulfilled or violated expectation resulting from a given rule. We therefore overload these operators and define alternative versions in which the last two arguments are omitted due to an implicit existential quantification:

$$\mathsf{Exp}(\lambda, \rho) \equiv \exists n,\phi\, \mathsf{Exp}(\lambda, \rho, n, \phi)$$
$$\mathsf{Fulf}(\lambda, \rho) \equiv \exists n,\phi\, \mathsf{Fulf}(\lambda, \rho, n, \phi)$$
$$\mathsf{Viol}(\lambda, \rho) \equiv \exists n,\phi\, \mathsf{Viol}(\lambda, \rho, n, \phi)$$

Using these operators and the model checker described previously [18] we can now analyse an observed execution trace to check for the activation, fulfilment or violation of expressive temporal rules of expectation, or, as special cases, more restricted representations used in prior work. For example, fulfilment of an obligation $O(\rho \leq \delta)$, stating that condition ρ must be brought about before deadline proposition δ becomes true [12], can be represented as $\mathsf{Fulf}(\top, \neg\delta\, \mathsf{U}\, (\rho \wedge \neg\delta)))$.

Note that we could also introduce additional versions of the Exp, Fulf and Viol operators that existentially quantify over only n or only ϕ; however in the remainder of this paper we will focus on the two-argument versions of the operators.

3.2 Nesting Expectation Operators

In the previous account of the logic, the Exp, Fulf and Viol operators could not contain nested occurrences of these operators. In this paper we allow this nesting, and explain why the previous restriction was unnecessary.

It follows from the definitions given above that the truth of the two versions of the Exp, Fulf and Viol formula do not depend on any future states in the model. This is because they depend only on the truth of a nominal (a special type of proposition) in the current state, Exp and Progress formulae in the prior state, and formulae prefixed by the $\mathsf{Trunc}_\mathsf{S}$ operator in the current and prior states. Formula progression, by definition, does not depend on future states, and the $\mathsf{Trunc}_\mathsf{S}$ operator eliminates them from consideration. Therefore it is meaningful for Exp, Fulf and Viol operators to appear within a condition of a rule (the first argument) inside one of these operators; the use of $\mathsf{Trunc}_\mathsf{S}$ to evaluate rule conditions will work correctly. For example, suppose that a library application has the rule of expectation $book\_borrowed \Rrightarrow \bigcirc book\_returned$ (where each state represents a day). Suppose that we also have a contrary-to-duty rule [13] of the form $\mathsf{Viol}(book\_borrowed, \bigcirc book\_returned) \Rrightarrow fine$, indicating that failure to return a book on time results in a fine (or, more precisely, in the expectation that a fine be imposed). In order to evaluate the formula $\mathsf{Exp}(\mathsf{Viol}(book\_borrowed, \bigcirc book\_returned), fine)$ we need to check whether in the current or any previous state a book was borrowed, and whether this book was returned on time or not. The key point is that in order

to evaluate the nested Viol formula, we never need to consult any future timepoints, due to the use of $\mathsf{Trunc}_{\mathsf{S}}$ in the semantics of Exp and Viol.

Additionally, to appear as the expectation of a rule (the second argument) within one of these operators, a formula must be able to be progressed when required—see the second line of the definition of the four-argument Exp operator. The axioms defining the progression operation that were defined previously [18] include the following base cases[5] (adapted slightly here for simplicity of presentation):

$$\mathcal{M}, i \models \mathsf{Progress}(\phi, \top) \text{ if } \mathcal{M}^i, i \models^+ \phi$$
$$\mathcal{M}, i \models \mathsf{Progress}(\phi, \bot) \text{ if } \mathcal{M}^i, i \models^+ \neg\phi$$

where $\mathcal{M}$ is a model, i is the index of a state in the model, and $\mathcal{M}^i$ denotes the model with all states after index i removed.

As Exp, Fulf and Viol can be evaluated without using any future states in the model, then one of the two base cases above will apply, and formulae having these operators as their principal functor will progress to either $\top$ or $\bot$. Therefore, these operators can also appear nested within the second arguments of these three types of formula and the restriction on nesting Exp, Fulf and Viol imposed in our previous work is unnecessary. Finally, the model checking process described in our earlier work [18] can be easily extended to apply to nested expectations, and we have extended our tool to be able to do so (as we will demonstrate in Section 5).

4 Use Cases for Nested Expectation Operators

In the previous section we argued that the restriction in previous work which did not allow expectations to be nested was unnecessary, and that the semantics of nested expectations are well defined and unproblematic. We also argued that checking whether nested expectations hold, are fulfilled or are violated, can be easily done within the existing framework and tool [18].

In this section we argue that allowing for nested expectations allows for a range of scenarios to be easily specified. Since we are making the case that nested expectations provide additional expressivity that is useful in a broad range of cases, we provide a number of different use cases in which nested expectations are used to specify desired normative behaviour. Space limitations prevent us from developing each of the scenarios in detail, but the aim is not to provide details on any given case, but, rather, to argue that a wide range of scenarios exists where there is a benefit from allowing nested expectations in a declarative way.

Chained expectations. One use case scenario for nested expectations is to allow for causality relationships between expectations to be captured. In this case, we may want to specify that a certain expectation ω exists when some other expectation has been fulfilled. We can express this as follows:

$$\mathsf{Fulf}(\phi, \psi) \Rrightarrow \omega$$

[5] Other axioms for Progress (not shown here) define $\mathsf{Progress}(\phi, \psi)$ compositionally based on the principal functor of ϕ and involve recursive progression of the top-level subformulae of ϕ.

In other words, once rule $\phi \Rrightarrow \psi$ is fulfilled, ω is expected.
This sequential fulfilment of expectations could arise when the two commitments must be fulfilled in a certain order due to one setting up the conditions for the other to be attempted, or when an agent's responsibilities are escalated as a result of successful performance.

Fulfilment ends probationary period. Another scenario, which is complementary to the one above, is that an expectation ω exists (only) until another expectation is fulfilled:

$$\top \Rrightarrow \omega \; \mathsf{W} \, \mathsf{Fulf}(\phi, \psi)$$

where W is the "weak until" operator: $\alpha \mathsf{W} \beta \equiv (\alpha \mathsf{U} \beta) \vee \Box \alpha$. In other words, ω is (unconditionally) expected until rule $\phi \Rrightarrow \psi$ is fulfilled, or, if the rule is never fulfilled, it is always expected.
This encodes the situation where some condition applies (e.g. a requirement not to operate some equipment without a qualified supervisor) until an agent ends a probationary period by fulfilling a certain expectation (such as passing a test).

"Contrary to duty" expectation or expectation to act on violation. Whereas the previous two cases dealt with the fulfilment of expectations, and how fulfilment may specify the termination or creation of another expectation, this rule deals with violation, and how it may result in the creation of another expectation:

$$\mathsf{Viol}(\phi, \psi) \Rrightarrow \omega$$

In other words, when rule $\phi \Rrightarrow \psi$ is *violated*, ω is expected.
This type of rule represents the well known concept of a contrary to duty expectation: if one expectation is violated an alternative expectation is created [13]. If ω involves a different agent to the one that violated the first expectation, this can represent the requirement for that agent to respond to the violation. A concrete example of this case that we discussed earlier is the expectation that a fine be imposed should a library book not be returned on time.

Just-in-time expectation management. The following form of rule could be used to encode a policy that resources for fulfilling a given expectation should be made available while that expectation is active.

$$\mathsf{Exp}(\phi, \psi) \Rrightarrow \omega$$

In other words, ω is expected while an expectation is active due to rule $\phi \Rrightarrow \psi$.

Delaying rule activation. The next few scenarios show how nested expectations can be used to specify constraints on the *timing* of expectations. The following rule expresses the policy to avoid the conditions that trigger an expectation until appropriate resources are in place for fulfilling it ("ω").

$$\top \Rrightarrow \neg \, \mathsf{Exp}(\phi, \psi) \, \mathsf{U} \, \omega$$

In other words, avoid triggering the rule $\phi \Rrightarrow \psi$ until ω is true. When ϕ does not refer to the future, this is equivalent to the simpler rule $\top \Rrightarrow \neg \phi \, \mathsf{U} \, \omega$. However,

this formulation makes the reason for the policy explicit: to avoid having ψ as an expectation.

Delaying rule fulfilment. Similar to the previous example, this is a constraint on the timing of an expectation relative to ω, but here we are specifying that the agent should not *fulfil* the expectation until ω. This may be desirable if, for example, an agent has a policy to not be over-diligent in fulfilling an expectation, e.g. it might only pay bills on the last possible day for payment.

$$\top \Rrightarrow \neg\, \mathsf{Fulf}(\phi, \psi)\, \mathsf{W}\, \omega$$

In other words, avoid fulfilling the rule $\phi \Rrightarrow \psi$ until (and only if[6]) ω becomes true.

Expectation handling priority. This is a special case of the previous use case. The following rule expresses a priority between two expectations:

$$\top \Rrightarrow \neg\, \mathsf{Fulf}(\phi, \psi)\, \mathsf{W}\ \mathsf{Fulf}(\lambda, \rho)$$

In other words, rule $\phi \Rrightarrow \psi$ should not be fulfilled until rule $\lambda \Rrightarrow \rho$ has been fulfilled. This could be used to express a policy for placing a priority on the order of fulfilment of rules.

Avoid violation between two states. Finally, this and the subsequent scenario deal with constraints over a time *interval*. Given two nominals, n_1 and n_2 we can specify that in the interval defined by the two end points a given condition must hold. For example, we may require that within a designated time interval a certain expectation should not be violated:

$$n_1 \Rrightarrow \neg\, \mathsf{Viol}(\phi, \psi)\, \mathsf{U}\, n_2$$

In other words, avoid violating the rule $\phi \Rrightarrow \psi$ between the states referenced by the nominals n_1 (inclusive) and n_2 (exclusive).
This example may be used in situations where an agent may be willing to risk violation of an expectation, but not during certain periods (e.g. when the boss is in the office).

Fulfil a rule sometime between two states. Similarly to the previous scenario, within a given time interval we can specify a condition, in this case that something (such as an expectation being fulfilled) *should* happen:

$$n_1 \Rrightarrow \neg n_2\, \mathsf{U}\, (\neg n_2 \wedge \mathsf{Fulf}(\phi, \psi))$$

In other words, the rule $\phi \Rrightarrow \psi$ must be fulfilled sometime between the states referenced by the nominals n_1 (inclusive) and n_2 (exclusive). The formalisation can be paraphrased as once n_1 has occurred, n_2 cannot occur until after $\mathsf{Fulf}(\phi, \psi)$.
This example may be used in situations where an agent subject to an expectation may adopt the policy of fulfilling it in a more restricted period than was originally required, e.g. while the boss is in the office and able to directly observe the fulfilment.

[6] As before, W is the "weak until" operator: $\alpha \mathsf{W} \beta \equiv (\alpha \mathsf{U} \beta) \vee \Box\alpha$.

5 Monitoring Nested Expectations through Model Checking

In this section we briefly illustrate that our model checker [18], extended with the ability to progress expectation modalities, enables nested expectations to be monitored. Although our model checker has an online mode, allowing states to be added and checked incrementally, for simplicity of presentation we illustrate the offline mode in which a complete model is checked.

We use the last use case above as an example. Consider the rule $p \Rrightarrow \Diamond q$ (once p holds, q is expected to hold eventually). We can encode the property that this rule will be fulfilled using our Python-based model checker as follows, where formulae are written in prefix form as nested tuples.

```
f = Formula(('Fulf', 'p', ('U', True, 'q')))
```

Now consider a model with four states $s_0, \ldots, s_3$ in which p holds in state s_1 and q holds in state s_3. We encode this as follows, where the first argument (4) is the number of states in the model and the second argument maps each proposition to a list of indices of states in which it holds, e.g. `'p':{1}` indicates that proposition `p` holds in state 1.

```
m = Model(4,{'p':{1}, 'q':{3}})
```

After invoking the model checker on this formula and model we can examine the value of property `f.labels` to find that the fulfilment formula `f` is only satisfied in state s_3 (we simplify the data structure to suppress details of the label structure not relevant to this paper):

```
{0: False, 1: False, 2: False, 3: True}
```

We now modify the formula so that our policy is to only fulfil the original rule from s_1 onwards and before s_3:

```
f = Formula(('Fulf', 's1',
               ('U', ('not', 's3'),
                 ('and', ('not', 's3'),
                   ('Fulf', 'p', ('U', True, 'q'))))))
```

This definition for `f` is simply the Python-based encoding of the formula:

$$\mathsf{Fulf}(s_1, \neg s_3 \,\mathsf{U}\, (\neg s_3 \wedge \mathsf{Fulf}(p, \Diamond q)))$$

which is the last use case scenario ("Fulfil a rule sometime between two states").

Invoking the labelling function again and examining `f.labels` we now find that the formula is false everywhere:

```
{0: False, 1: False, 2: False, 3: False}
```

This is the expected result as the new formula can only possibly be satisfied in states s_1 and s_2, and the original formula does not hold in those states.

6 Conclusion

In this paper we have considered the ability of approaches for modelling and monitoring various sorts of expectations to represent the existence, fulfilment and violation

of expectations as first-class entities, and to allow these to appear nested within rules of expectation. Having found no work in the literature that fully meets these needs, we demonstrated how our existing approach to modelling and monitoring expectations extends readily to address this issue. We listed some use cases to show that the expectations expressible using this new modelling ability are of interest in practical settings.

At present our focus is on passive detection of expectation creation, fulfilment and violation (i.e. monitoring). However, as our use cases included several examples of policies that agents might adopt, we need to investigate ways in which agents informed by these expectations can understand them and proactively adjust their behaviour to fulfil such policies. Specifically, an agent could use the model checker to reason about hypothetical extensions of the history so far. Given a history H, and an agent who is considering either action A_1 (resulting in state S_1) or action A_2 (resulting in state S_2), then we could use the model checker to label the extended history $H \oplus S_1$ (where "$\oplus$" denotes sequence concatenation) and the extended history $H \oplus S_2$, and use the results to guide decision making by the agent. More generally, an agent might realise via analysing traces that its current expectation doesn't meet some soft constraint (e.g. getting praised by the boss), and therefore follow a heuristic or apply some reasoning technique to restrict the period in which it aims to satisfy the expectation (e.g. so that the boss will witness it). To allow this we must also extend our framework to allow the content of expectations to identify which agents are responsible for particular expectations. Finally, the fulfilment operator (and other operators) could be extended to allow not just a single rule to be given as an argument, but a *set* of rules.

References

1. Ågotnes, T., van der Hoek, W., Tennenholtz, M., Wooldridge, M.: Power in normative systems. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 145–152. IFAAMAS (2009)
2. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the $\mathcal{S}$CIFF framework. ACM Transactions on Computational Logic 9(4), 29:1–29:43 (2008)
3. Alberti, M., Gavanelli, M., Lamma, E., Chesani, F., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based software tool. Applied Artificial Intelligence 20(2), 133–157 (2006)
4. Aldewereld, H., Álvarez-Napagao, S., Dignum, F., Vázquez-Salceda, J.: Making norms concrete. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems, pp. 807–814. IFAAMAS (2010)
5. Artikis, A., Sergot, M.: Executable specification of open multi-agent systems. Logic Journal of the IGPL 18(1), 31–65 (2010)
6. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artificial Intelligence 116(1-2), 123–191 (2000)
7. Bentahar, J., Moulin, B., Chaib-draa, B.: Commitment and Argument Network: A New Formalism for Agent Communication. In: Dignum, F. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 146–165. Springer, Heidelberg (2004)
8. Bentahar, J., Moulin, B., Meyer, J.J.C., Lespérance, Y.: A New Logical Semantics for Agent Communication. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA 2006. LNCS (LNAI), vol. 4371, pp. 151–170. Springer, Heidelberg (2007)

9. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2001)
10. Boella, G., van der Torre, L.: Security policies for sharing knowledge in virtual communities. IEEE Transactions on Systems, Man and Cybernetics, Part A 36(3), 439–450 (2006)
11. Boella, G., van der Torre, L., Verhagen, H.: Introduction to the special issue on normative multiagent systems. Autonomous Agents and Multi-Agent Systems 17(1), 1–10 (2008)
12. Broersen, J., Dignum, F., Dignum, V., Meyer, J.J.C.: Designing a Deontic Logic of Deadlines. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 43–56. Springer, Heidelberg (2004)
13. Carmo, J., Jones, A.J.I.: Deontic logic and contrary-to-duties. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 8, pp. 265–343. Kluwer Academic Publishers (2002)
14. Castelfranchi, C.: For a systematic theory of expectations. In: Proceedings of the 2nd European Cognitive Science Conference. Taylor & Francis (2007)
15. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the reactive event calculus. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), pp. 91–96. Morgan Kaufmann (2009)
16. Chittaro, L., Montanari, A.: Efficient temporal reasoning in the cached event calculus. Computational Intelligence 12(3), 359–382 (1996)
17. Cliffe, O., De Vos, M., Padget, J.: Modelling Normative Frameworks Using Answer Set Programing. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS (LNAI), vol. 5753, pp. 548–553. Springer, Heidelberg (2009)
18. Cranefield, S., Winikoff, M.: Verifying social expectations by model checking truncated paths. Journal of Logic and Computation 21(6), 1217–1256 (2011)
19. Dastani, M., Brandsema, J., Dubel, A., Meyer, J.-J.C.: Debugging BDI-Based Multi-Agent Programs. In: Braubach, L., Briot, J.-P., Thangarajah, J. (eds.) ProMAS 2009. LNCS, vol. 5919, pp. 151–169. Springer, Heidelberg (2010)
20. Dignum, F., Meyer, J.J.C., Wieringa, R.: A dynamic logic for reasoning about sub-ideal states. In: Proceedings of the ECAI Workshop on Artificial Normative Reasoning, pp. 79–92 (1994)
21. Dignum, F., Weigand, H., Verharen, E.: Meeting the Deadline: On the Formal Specification of Temporal Deontic Constraints. In: Michalewicz, M., Raś, Z.W. (eds.) ISMIS 1996. LNCS (LNAI), vol. 1079, pp. 243–252. Springer, Heidelberg (1996)
22. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Van Campenhout, D.: Reasoning with Temporal Logic on Truncated Paths. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 27–39. Springer, Heidelberg (2003)
23. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 236–243. IEEE Computer Society (2004)
24. Farrell, A.D.H., Sergot, M.J., Sallé, M., Bartolini, C.: Using the event calculus for tracking the normative state of contracts. International Journal of Cooperative Information Systems 14(2 & 3), 99–129 (2005)
25. Fornara, N., Viganò, F., Colombetti, M.: Agent communication and artificial institutions. Autonomous Agents and Multi-Agent Systems 14(2), 121–142 (2007)
26. Gabbay, D.: The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 409–448. Springer, Heidelberg (1989)
27. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: Implementing norms in electronic institutions. In: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 667–673. ACM Press (2005)

28. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.W.: Constraint rule-based programming of norms for electronic institutions. Autonomous Agents and Multi-Agent Systems 18(1), 186–217 (2009)
29. Governatori, G., Rotolo, A.: How do agents comply with norms? Dagstuhl Seminar Proceedings 09121 (2009), http://drops.dagstuhl.de/opus/volltexte/2009/1909
30. Lomuscio, A., Penczek, W., Solanki, M., Szreter, M.: Runtime monitoring of contract regulated web services. In: Proceedings of the Third Workshop on Formal Languages and Analysis of Contract-Oriented Software, pp. 9–16 (2009), http://www.dsi.uclm.es/retics/flacos09/Flacos09-proceedings.pdf#page=9
31. Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 153–160. IFAAMAS (2009)
32. Nickles, M., Rovatsos, M., Weiss, G.: Expectation-oriented modeling. Engineering Applications of Artificial Intelligence 18, 891–918 (2005)
33. Raimondi, F., Lomuscio, A.: Automatic Verification of Deontic Properties of Multi-agent Systems. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 228–242. Springer, Heidelberg (2004)
34. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
35. Singh, M.P.: Semantical considerations on dialectical and practical commitments. In: Cohn, A. (ed.) Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 1, pp. 176–181. AAAI Press (2008)
36. Spoletini, P., Verdicchio, M.: An Automata-Based Monitoring Technique for Commitment-Based Multi-Agent Systems. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 172–187. Springer, Heidelberg (2009)
37. Tinnemeier, N.A.M., Dastani, M.M., Meyer, J.J.C., van der Torre, L.: Programming normative artifacts with declarative obligations and prohibitions. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 145–152. IEEE Computer Society (2009)
38. Vázquez-Salceda, J.: The role of norms and electronic institutions in multi-agent systems applied to complex domains: The HARMONIA framework. AI Communications 16(3), 209–212 (2003)
39. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Implementing Norms in Multiagent Systems. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 313–327. Springer, Heidelberg (2004)
40. Verdicchio, M., Colombetti, M.: Communication languages for multiagent systems. Computational Intelligence 25(2), 136–159 (2009)
41. Wallace, I., Rovatsos, M.: Bounded practical social reasoning in the ESB framework. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 1097–1104. IFAAMAS (2009)
42. Yolum, P., Singh, M.P.: Commitment Machines. In: Meyer, J.-J.C., Tambe, M. (eds.) Intelligent Agents VIII. LNCS (LNAI), vol. 2333, pp. 235–247. Springer, Heidelberg (2002)
43. Yolum, P., Singh, M.: Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and Artificial Intelligence 42, 227–253 (2004)

Operationalization of the Sanctioning Process in Utilitarian Artificial Societies

Tina Balke[1,2] and Daniel Villatoro[3]

[1] University of Surrey, Centre for Research in Social Simulation
[2] University of Bath, Dept. of Computer Science
t.balke@surrey.ac.uk
[3] Artificial Intelligence Research Institute (IIIA), Spanish Scientific Research Council (CSIC)
dvillatoro@iiia.csic.es

Abstract. With the advent of highly distributed and populated artificial societies where centralized coordination is unfeasible, normative multiagent systems have moved into the focus of attention – as they are promising for improving agent interactions and minimize social frictions. However, an important point that deserves to be studied in detail is what happens when agents behave egoistically and possibly violate the norms they should comply with. The objective of this work is to present an integrated view of the *sanctioning process* and analyze each of its phases with regard to its operationalization in artificial societies. Moreover we review several sanctioning mechanisms presented in the multiagent literature and examine them in the context of our proposed process.

1 Introduction

Crime and punishment have been a constant in human affairs for as long as love and greed, or feed and fight; and have moved the will of men, and men have found the need to control that will for the benefit of society. Likewise, in order to function well, artificial and technologically enabled societies are finding it necessary to establish regulatory frameworks and ways to enforce them.

Enforcement is concerned with trying to ask (and answer) what happens if an agent does not comply with the obligations or prohibitions of a normative system as well as how to react to that. As stated by Ågotnes et al. [1] or Coleman [13], for this reason *compliance* is one of the most important issues associated with normative systems. In this paper we will focus in detail on one aspect of enforcement: we will analyze sanctioning[1] as one device to motivate normative compliance[2].

The main motivation of this work is to build a complete map of the sanctioning process. Thus, although many papers have dealt with sanctioning already (e.g. see the works mentioned in Sec. 4), to the best of our knowledge, the works existing to date

[1] Despite one of the authors elsewhere [39] differentiated between sanction and punishment, for the scope of this work both words are considered synonyms and refer to the same process.

[2] For us, enforcement is a process and sanctioning is one possible specification of this enforcement process. Another enforcement specification – which we do not focus on in this paper – could for example be to reward positive behaviour.

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 167–185, 2012.

do not attempt to give an overview of the design options available in the sanctioning process or the roles involved in it. Instead, they normally fix the restrictions necessary for their environment and focus on a particular mechanism or framework. These restrictions of the environment and the assumptions made about the assignment of roles, as well as the different foci of the analyzed works, make the information on these design options very dispersed. As a consequence it is difficult to get an overview of the choices that can be made when designing a sanctioning mechanism.

We try to close this gap by presenting a comprehensive overview on the design options and roles relevant when talking about sanctioning. We do not intend to reinvent the wheel. On the contrary, our first aim is to to elucidate the relevant facets of sanctioning in this context, establish pertinent conceptual distinctions and eventually provide a coherent framework so that available complementary developments may be brought together and missing pieces become evident. Moreover, as we indicated above, we have a second aim which is to make practical use of sanctioning as a motivational device for normative compliance in actual MAS.

As mentioned before, to us sanctioning, being one means of enforcement, is a process. This process is composed of different phases that – although interlinked – can be designed with a degree of independence. That is why in this paper we will analyze the *sanctioning process* and the phases it consists of. By identifying all the features in each of the phases that compose the sanctioning process, we aim at obtaining a general view of the overall process.

In order to arrive at our overall sanctioning process, in the next section, an introduction to the general state of the question and the literature associated with enforcement and sanctioning will be given. Furthermore, we delineate our object of study, make our simplifying assumptions explicit and set our work within the context of the research domain. One of the assumptions is that we think about utilitarian agents. Thus, emotions, despite playing a significant role in the sanctioning process, will not be investigated at this stage[3]. In Sec. 3 we then study the sanctioning process as a sequence of phases and particular components that we believe deserve serious ulterior formal treatment. Next, in Sec. 4, we take a step towards our second aim and do a first matching exercise against five well documented MAS platforms for regulated open MAS. We close this paper with an outline of our current and future work.

2 State of the Question

Literature associated with the notion of sanctioning is overwhelming. Two traditional approaches, moral and legal, over centuries of study and practice, have established not only a rich conceptual universe around the notion of sanctioning but innumerable devices to address its practical aspects as well [33]. We will draw inspiration from these sources; however we will base our proposal on works that are closer to the MAS world.

A crude classification of sanctioning-related MAS literature may end up with four major blocks, which will now be explained in more detail.

In the first block it is assumed that the actors (and/or their actions) in a system can be controlled (this includes the control of the physical power of actors) and any

[3] Instead we refer the interested reader to Scheve et al. [36] as well as Staller and Petta [37].

non-normative actions can be stopped before they can take place and affect the system. This notion is typically referred to as regimentation [28] or control-based enforcement (CBE) [35, p. 14]. Regimentation tries to bring about a state in which the deviation from a norm is physically not possible, or where the deviation from the norm does not have any effect on the rest of the participants in a system. Minsky [33] distinguished two modes of regimentation which together comprise the regimentation idea: regimentation by *interception* and regimentation by *compilation*.

Regimentation by compilation assumes that all participants' mental states are accessible to the system (closed systems), and can be altered to be in accordance with the norms effecting the system. Thus, participants are treated as a "white box" whose content can be analysed and altered as needed by the regimenting entity of the system [3]. Minsky gives an example of a computer program component that is supposed to join a running system. In his example the source code of this new program is accessible to the regimenting entity of the system and this entity can check whether based on the source code the program can conduct any actions that deviate from the norms of the system. If this is the case it can either stop the program from joining the system or urge a change to the program. This concept is applied for example in the KAoS architecture [7].

In the case that the mental states are not accessible to the system (i.e. the inner states of the participants are a black box to the system) norm compliance is ensured by indirectly constraining the actions of the individual participants. Thus, regimentation by interception uses a regimentation component (typically a piece of middleware in computer systems) that at run-time intercepts all actions (in computer systems this normally refers to messages being sent to the rest of the components) and dismisses those that are not in accordance with the norms of the system. Looking at the implementation side of interception-based regimentation, in artificial societies, according to [35, p. 15] this approach is relatively easy to implement and deploy. Nevertheless it exhibits a number of problems which act as drawbacks to the idea and will be discussed later in this paper.

There is a second block of Economics-inspired works that have studied sanctioning as incentive for rational behavior of utility-based agents. In this area, sanctioning is seen as an amount taken from an agent's benefits and the effectiveness of sanctioning is usually measured against system equilibria, in line with the theory of Becker [5]. The topic has been framed mostly in terms of mechanism design and the issues that economists have studied more thoroughly are the information about infraction and sanctioning [25], as well as the amount and pervasiveness of sanctioning [18]. Methodology has been either game-theoretic (see [13] for example) or experimental (e.g. [29]). In this latter case classical experimental economics methods have been enriched with agent-based simulation and thus explore situations that are difficult to examine with set-ups involving human subjects only [8][4].

[4] One group of mechanisms which is sometimes seen separately, but which we bring under the category of economic-inspired approaches, are reputation mechanisms. We view reputation mechanisms as economic approaches, based on the rationale, often found in reputation mechanisms, that a bad reputation leads to future negative utilities (e.g. if no cooperation partners can be found due to a bad reputation), and under the rational utility maximizing assumption economic considerations are made. Alternatively, if one considers the psychological aspects of reputation, it could also be placed in the category of cognitive-inspired solutions.

A third block is inspired by formal normative concerns, as they apply to agents and MAS. Some works deal with sanctioning, and incentives in general, as a component of the notion of a norm and thus study the structural relationships of those components. Issues addressed include the relationship between target and victims[5], the syntax of activation and deactivation conditions or links between infractions and reparatory actions [34]. Others are interested in the dynamics of norm-compliance and thus deal with sanctioning as events triggered when an infraction occurs [27]. Finally there are works that take sanctioning as a feature that depends on the type of norm (conventions, social, regimented, functional,... [31]). These are interested, for instance, in the operational semantics of compliance and enforceability [28], or in the class of incentives most naturally associated with different norm types. Works in this block range from the strictly formal to the mostly computational approaches.

The fourth block is composed of works that are concerned with the cognitive aspects of punishment. Some works are interested in the role of sanctioning in the adoption or internalization of norms [14], others in the role or modes of learning and adaptation that may be associated with sanctioning [2], and yet others in the way different ways of sanctioning being more or less effective depending on the cognitive "make-up" of agents [2].

Our work will be based on knowledge from the three blocks and for the purpose of this paper – since our purpose is merely exploratory here – we will use that common knowledge without specific reference.

Our first task will be to state our subject matter with some precision. That is impose some simplifications to the sanctioning process in open MAS, so we will state our intended meaning of some terms and narrow down the scope of the discussion.

2.1 Regulated Open MAS

Objective MAS. Technically speaking we assume the existence of a MAS environment –observable as an objective entity– with fixed ontology, dynamic state, regimented constitutive conventions and participation of capable and entitled agents.[6]

Open MAS. The decision-making processes of agents may be not under the control of the MAS. Agents may respond to their own motivations and be owned by different owners. Agents may enter or leave the MAS at will, as long as they are capable and entitled to be in the MAS.

[5] A detailed description of our understanding of the terms is given in Sec. 2.3.

[6] That is (i) there exists a *world* that may be populated by (ii) *agents* whose activity may involve other agents and possibly other entities – like speed limits, traffic lights, or fines. (iii) At any point in time, that world is in a *state* which is represented by a set of variables (sometimes referred to as a trace [9]) whose values may change only due to the actions of agents. (iv) Out of the many conceivable actions which agents theoretically may attempt, only the ones that are deemed *admissible* in that world may have any effect in the world. (v) All actions are subject to preconditions (on the state of the world) that make them feasible and post-conditions that determine their intended effect in the world. (vi) Only agents that are capable of performing admissible actions and have the proper authorization may exist and act in the world.

Regulated MAS. Assuming: (i) Interactions are regulated by norms. (ii) Agents may decide whether they comply with those norms or not. (iii) Only observable behavior is subject to be regulated. (iv) Norms are intended to be enforced.

2.2 Norms

Norms are artificial restrictions on agent behavior. In principle, norms dictate what actions are permitted, prohibited or obligatory under given conditions as well as the effects of complying or not with those norms.[7]

Norm structures are the components that define a norm for the purpose of punishment. For the moment we assume the following: the *antecedent* determines the activation and termination conditions; the *consequence* specifies the intended behavior; the *target* agents are those that should abide by the norm; the *goal* indicates the purpose and beneficiaries; *associated norms* whose activation / deactivation is dependent on activation / deactivation of this norm; the *triggered actions* explains its effects and consequences (mostly sanctions and reparatory actions).

A **Normative system** is formed by (i) a regulated open MAS, (ii) the set of norms that regulate it, (iii) the conventions for the creation, issuance, diffusion and change of norms, (iv) a social structure of the MAS involving roles and relationships among roles, (v) appropriate governance mechanisms for norm-enforcement, (vi) performance criteria that determine the quality of the normative system.

Normative context is (i) a normative system, (ii) a set of participating agents and (iii) a non-empty set of states of the MAS.

Roles are a way of defining the tasks that differentiate groups of participants, and thus may be seen as entitlements and obligations of any agent that may execute a given collection of tasks. Hence, a normative system includes norms that define whether an agent may change roles or perform more than one role at a given time, what roles may be compatible or incompatible, and whether roles may have hierarchies and of what types.

Norm types in terms of sanctioning (see [40] for example): (i) Constitutive and regimented norms are impossible to violate (e.g., traffic lights have three colors). (ii) Conventions, should be observed —because otherwise some action would not be effective— but are not directly enforced and have no direct sanction associated with them (e.g., salute protocols). (iii) Functional, or regulative/enforceable norms should be observed and if violated some corrective action may ensue.

2.3 Governance

The act of a system designer or governing entity in making decisions that specify rules, defining expectations, granting power, or verifying performance in order to steer the system in a desired direction is called governance.

[7] In general, norms are *explicit*, although norms may indeed be an emergent feature of a normative context and sanctioning a means for turning implicit disposition of individuals into norms to be observed in society as a whole.

The word governance itself is derived from the greek verb $\kappa\upsilon\beta\epsilon\rho\nu\alpha o$ (*kubernáo*) which means "to steer". Governance may be done in any system of any size ranging from individuals to enterprises to complete nations.

One part of governance is the observation of normative compliance, both with respect to the feasibility of compliance as well as with respect to enforcing compliance of the system participants (i.e. agents). As pointed out earlier, in this paper we review sanctioning as one means of supporting compliance.

Sanctions and Punishment. Non-compliance may involve some penalty on transgressors and, compliance some rewards. We will only refer to sanctioning in this paper, although *mutatis mutandis*, the framework applies to both –to rewards and to sanctions– in general. Only prescribed and perceivable behavior may be sanctioned.

Type of Sanctions. Following [38], we recognize two: *direct* and *indirect.* Direct sanctions affect the agent immediately and are noticeable directly (like bans, fines and physical punishment). Indirect sanctioning affects only the agent's future actions and may be ostensible or not (e.g. warnings, reputation). One interesting feature of indirect sanctions, such as reputation, is the subsumption of roles involved: by spreading rumors about a *violator*, every agent becomes a *judge* (as any agent can reinterpret the rumor in every way it perceives to be the correct way) as well as an *executor* (any agent may use the rumor in what ever way it wants, and redistribute it to whom ever it wants). Hence, indirect sanctions might be irreversible, uncontrollable and unmeasurable, obtaining (in a worst case scenario) an infinite sanction against the *violator*.

Objective of Sanctions. While the general purpose of sanctioning is to motivate norm compliance, we may distinguish different effects such sanctions are intended to have on future behavior. Customary objectives are deterrence, compensation, retaliation and, finally, exemplification and learning.

Utilitarian agents will be used in this paper for illustration purposes. In the tradition of Hedonism we assume that all agents try to maximize their own utility – hence we refer to utilitarian artificial societies in the title. As a result – in accordance with neoclassical economic tradition – for the agents we assume that their goals, motivations and decision-making processes take into account many issues that may be amalgamated into a single value —*utility*— which is computable through a "utility function" [26]. Thus any sanction will be associated with an increment or a decrement of the utility of transgressor and victims. Furthermore, we will assume that utility functions may be different for different individuals plus the existence of a (general) utility of the system. For utilitarian agents, therefore, sanctions are always financial penalties. In their case, the value of indirect penalties, like reputation, usually cannot be calculated.

Non-compliance qualifications allow the possibility of distinguishing degrees and gravity of non-compliance of some norms and, consequently, modes and intensity of their sanctions.

Governance roles involve entitlements and capabilities of MAS participants to detect and evaluate transgressions, and to impose sanctions and enact reparatory actions and, in some normative systems the possibility of adding or modifying norms (e.g. *legislator*). Some normative systems assume complete institutional governance (all governance is in the power of the objective MAS), while others delegate governance in

specialized agents. Some systems are mostly self-regulated and still others may have only the governance that each participant may grant to its own actions.

Two basic roles (with regard to norms) have already been identified [13]: the beneficiaries and the targets. *Targets* are the actors for whom the norm is specified for. *Beneficiaries* are those actors who benefit from the norm and potentially hold the norm. However, some roles are still missing with regard to governance: the violators, the victims, the profiteers, the observers, the judges, the executors, the controllers and the legislators. *Violators* are the actors that have taken a different action than the one specified by the norm. *Victims* are the actors who suffer the consequences of a norm violation. *Profiteers* are the actors who benefit from the consequences of a norm violation. *Observers* are those actors that identify a violator. *Judges* are those actors that given the information about the violation are able to calculate the sanction to be applied to the violator. *Executors* are the actors that apply the sanction to the violators. *Controllers* are the actors that –after the sanction has been applied– control that the sanction had the desired effect. *Legislators* are the actors that observe the efficiency of the system with respect to the fulfillment of norms and application of sanctions.

In some cases an actor might assume more than one of these roles. The combination of roles that each actor possesses is completely related with the type of type of society in which agents are are located, and vice-versa. A society where all the roles are played by different agents is completely different to a society where all the roles are played by all agents.

Normative System Dynamics. While the manifest purpose of sanctioning is to motivate compliance, a collateral effect is the adaptation of behavior of individuals to the normative context and in a deeper level, the adaptation of the normative system to an evolving normative context. Thus the cost and effectiveness of sanctioning may induce changes in the sanctioning process in order to better achieve its goals, and may eventually call for a modification of the whole normative system. Both situations should be contemplated in the governance devices of the regulated MAS and, depending on the system design, they may be embedded in the system or handled off-line.

3 The Sanctioning Process

As shown in Fig. 1 we conceptualize sanctioning as a four-stage process. The first three stages correspond roughly to the conventional processes of arrest, trial and conviction of transgressors, while the fourth is the process of learning and adaptation that ensues. Each stage involves distinctive activities whose performers are agents playing particular roles. Although in general terms most activities and roles are present in every regulated MAS that includes enforceable norms, they need to be adapted to the particularities of the normative context where violations take place. Here we will sketch the general contents but limit occasional comments to utilitarian agent sanctioning.

3.1 Detecting Non-compliance

This process has two goals: the ascertainment of a violation and the identification of agents involved. Two obvious roles take part in this phase: *violator* and *observer*.

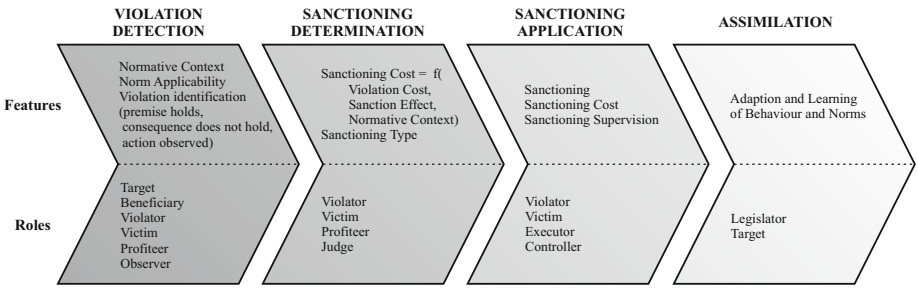


Fig. 1. The Sanctioning Process

One should note that, depending on the structure of the norm that is violated, the observer may need to gather enough evidence to ascertain what violation actually took place before any further punitive actions can take place. Hence, in order to bring about charges, observers may need to assess *damages*, to assign *blame* and to identify *victims* and *profiteers* that may be affected by the non-compliance.

Each of these roles may be performed by more than one agent and, in totally self-regulated MAS, even performed by the same agent. In the particular case of utilitarian agents, the observer role may take different forms:

- A *first-party observer* who controls its own compliance.
- A *second-party observer* who observes a misbehavior of a transaction partner(s). An example would be the buyer on eBay, who has bought and paid for a product, but the seller does not send it to him. If the product was to be sent insured, he can recognize a violation by the seller if he does not receive a parcel on time.
- *Third-party observers* that supervise the behaviour of other agents in the system.
- *Institutional watchmen* are agents whose decision-making and action is controlled by the MAS and have as a goal the detection of transgressions and their proper assessment. Each one of these may have limited observation capabilities, as for example, the Second Life Governance Team [16], where a team of authority representatives patrol the grid, ensuring the correct behaviour of the citizens. If they are not present at the scene of a norm violation, then, they cannot detect it. Evidently, full observability may be achieved by placing watchmen in every scene, as in the case of staff agents in electronic institutions [19], who control every interaction.
- *Institutional enforcement* – in terms of regimentation or CBE discussed beforehand – is another way of achieving full observability of violations is by embedding it as a functionality of the MAS as a whole [28,6], however especially in open distributed MAS this approach is unfeasible due to the nature of the system, i.e. as neither the mental states required for regimentation by compilation seem accessible nor is it likely that one can observe all actions by all agents in open systems.

The question of whether all violations may be detected or not may determine strategic decisions of agents with respect to norm fulfillment. In scenarios where every violation is observed, agents would violate norms only if they consider that the benefits of a

violation are higher than the penalty. On the other hand, when violations might pass unnoticed, agent strategic behavior is also possible but then benefits are compared against expected (uncertain) penalties.

Besides the strategic behaviour of agents in case no complete observation is feasible, other questions arise in the violation detection phase. One major question is the identification of the target agents to which a norm can apply. Typically in order to violate a norm, the agents need to be targets of that norm, i.e. the norm needs to specify that what they are doing is a violation. When trying to observe violations it is therefore of great importance to determine whether the presumed violators are actually targets. To give an example: drinking strong alcohol is only allowed at the age of 18 in many countries. The restriction thus applies to all people younger than 18 who are the *targets* of this norm. When determining whether someone has violated the norm it is of importance, whether this person is below 18 and therefore a target. In open systems where not all information of agents are freely available this can be difficult and it can for example happen that situations are falsely identified as violations because of a lack of knowledge that the involved agents are no targets.

A second major problem in the detection phase is that norms can change and as a consequence, situations / actions that have been violations earlier are not violations any more (or earlier non-violations become violations). Disseminating norms as well as keeping track of times of violations (in order to determine whether a norm was applicable at the point of time of the violation) therefore should be considered in this phase.

3.2 Sanction Evaluation

This phase involves the appraisal of the applicability of the norm within the normative context of the non-compliant action and, if applicable, activation of the normative consequences of infringement and determination of the actual sanction to be imposed. In this stage, *observers* bring charges to a *judge* who should decide if the violator deserves a sanction and then determine the appropriate sanction. The determination of the sanction can for example depend on whether the "violating agent" was actually a target of the norm, whether the violation happened due to conflicting norms and the agent therefore had little chance on avoiding violation or whether the agent violated the norm intentionally or not (in case it was not aware of the norm). The latter of these considerations seems difficult to determine in open systems for the same reasons regimentation by compilation is difficult.

The judge may also command the execution of reparatory actions as a consequence of the infringement. In some regulated MAS, violators and victims may be party to argumentation processes to establish applicability and severity of sanctions.

Sanctions are usually calculated as a function of the violator, the victims, the effects produced by the violation, and the normative context where the violation was detected. This sanction calculation might imply a cost which is absorbed by the *judge*. Some may need to be enacted either automatically or commissioned by the judge.

During sanction calculation for utilitarian agents, one should not assume that all agents are regimented by the same utility function. Even though taking this factor into

account when calculating a sanction is crucial, this is unfortunately impossible, as normally the utility function of each agent is private (i.e. only known by the respective agent). In case this utility function was accessible, the sanctioning entities would be able to calculate a perfect sanction to decrement the utility function of the violator, even if they do not share that function. Consequently, to ensure the intended effect of a sanction we have to be aware of the agents' utility function, or at least, the *judges* and the *executors* agents do use a sanction that affects the utility of the *violator*.

One final decision has to be made before applying the sanction, and this is how the sanction should applied. The way a sanction is applied can be decided by the *judge* or be imposed by the normative context.

3.3 Sanction Application

The sanction application phase is composed of the actual execution of a sanction and the assessment of its proper application. The outcome of the previous stage is a sentence to be carried out, and the role that the respective agents perform in order to bring about the sanction is the *executor* role. Executors may be [3]:

- the *violator* itself, who for example after violating a norm (un)intentionally might want to repay the damage done (maybe to avoid a loss in reputation).
- A *first-party victim*: depending on the types of norms that are controlling the society, one agent can be directly harmed by another agent's norm violation. If this is the situation, the victim can act as the observer, judge and executor of the sanction to the violator. Agents being victims of a violation and deciding not to interact with the violator again in the future are one example of such a setting.
- A *third-party observer*: if an agent is seen violating a norm, the observing agent can have the right to apply a sanction to the violator, even if this agent did not suffer from the violation.
- A *group of third-party observers*: the act of sanctioning can be distributed amongst a group of agents. This type of sanctioning act is often used in indirect sanctions such as reputation (which has an aggregated effect, the more agents use it, the more powerful).
- *Institutional enforcers* such as authority agents who do not observe all actions but only the ones within their vicinity, and do have the power (designated by the institution that they all belong to) to apply the sanction.

Two problems typically occuring in the sanction application phase are (i) that in order to apply a sanction the executors might need power and permission to do so. A lack of these two might therefore result in the intention to apply a sanction whose execution is unsuccessful. (ii) If several agents perform the roles of executor, it might result in situations in which the agent that has violated the norm is sanctioned several times by the different executor agents, and not only once. Although this might be intentional (e.g. in reputation mechanisms), in the current legal considerations on direct sanctions the view that one should only be sanctioned once for a violation is taken.

Another aspect that needs to be considered is that sanctions may not come for free but may have a cost associated to their application that may be bore by *executors* or the

regulated MAS. Sometimes the cost is directly associated with the sanction. A straight forward example of *costly sanction* is imprisonment, where the state has to support jailmates. However, the application of other sanctions imply an unknown cost to the *executors* but still inflict damage to the *violator*. Reputation is normally one of the most effective *relativized-cost* sanctioning mechanism. Transmission of (bad) information regarding an agent has a relative cost (depending on the degree of desposability of the information transmitted and the retaliation level of the members of the society) to the agent transmitting the information, however, it might also affect the target of the rumor.

After the *executor* has acted, the *controller* is in charge of controlling that the *violator* has indeed received the sanction, and that the sanction has served the purpose for which it was designed. The *controller* is also responsible for ensuring that other actions that are associated with the infringed norm are properly triggered and carried out. In case of compensational sanctions, in particular, the *controller* will ensure that the *victims* are compensated.

3.4 Assimilation

Assimilation is the processes through which individuals or the normative system itself take advantage from a sanction to modify ulterior behavior.

As we have seen, by performing the corresponding roles, violators, observers, victims, judges, executors come in contact with information about norm compliance that they could ideally incorporate in their decision-making mechanism and may hence shape their own future behavior. This learning – typically by the norm *targets* – is the main focus in most systems that take into account normative learning. In addition to this learning by interaction, in a normative system other means of communicating learning-relevant information to the participants can be implemented. This facilitates that norms about punishment may give shape to a space for individual evolution where specific aspects of compliance are given more relevance than others and therefore facilitate evolution of the system along different lines.

While punishment is intended as a motivation for actions and the sanctions of specific specific norms may have an ostensible objective (compensation, retaliation, deterrence, exemplarity), the actual effect of punishment in an agent's motivations is a private matter of convictions, thus directly unobservable for the MAS. Nevertheless, the ulterior behavior of agents is observable, hence the MAS as a whole, or its *legislators*, may use evolution of the behavior of individuals as input for *purposeful* evolution of the normative system. Autonomic MAS would use evolution of individuals' behavior for *self-adaptation*. In either case, modifications of behavior need to be observable somehow. The natural means are probes and indicators that must be aligned with performance parameters accessible to legislators or the system dynamic features.

This way, the choice and balance of those conventions that determine the availability and form of information about punishment and norm-compliance have significant effects in the overall collective behavior[8].

[8] By collective behavior we refer to both, collective behaviour that results from the aggregate behavior of informed individuals, as well as collective behavior determined by new norms that result from legislative or autonomic adaptation.

4 Contrasting the Process

Having presented our sanctioning process in the last section, in this section we review some of the existing MAS sanctioning approaches in the context of our process (due to space constraints this review is not complete, although we believe it is representative). In the next paragraphs we will contrast the approaches in more detail. Table 1 gives an overview over the approaches contrasted.

4.1 eInstitutions [19,27]

The eInstitutions (eI) framework was first described in [22]. The framework consists of three components: (i) a formal specification of a normative system (called institution by the authors [23,21]), (ii) the ISLANDER tool for editing the specifications [20], and (iii) AMELI, a run-time environment for executing the system, based on the specifications [24].

With respect to institutions the eI specifications consist of formal semantics for the notion of an institution and its components (abstract and concrete norms, empowerment of agents, roles) and defines a formal relation between institutions and organizational structures. In the original design of the eI the sanctioning was done with the help of regimentation mechanisms. In the eI, all actions are triggered with the help of messages. These messages that trigger actions by the individual agents are observed by staff agents (that are implemented in the middleware of the system). The staff agents check every message and do not forward them further in case they violate a norm. This way normative compliance was unavoidable. As a consequence of this rigorous non-compliance identification, very little options for the remaining two phases exist. However, recent work [27] has started to include norm violations considerations, although these violations are detected by the institution, which is ready to apply the sanction to the violator.

4.2 OperA [17]

The OperA model is a framework for agent societies with the goal to legitimise the concept of autonomy between agent goals and society requirements. The model makes use of contracts that predefine the norms and their corresponding sanctions in case of violations. All the parameters of the contract are negotiated and are used to define the transaction partners the *targets*, *beneficiaries*, potential *violators*, and *judges*. Hence in the calculation phase of our model, existing well defined sanctions are assumed and do not have to be reasoned about. In order to verify that agents fulfill the contracts, Dignum introduces the idea of Trusted Third Parties (TTPs) that verify the compliance at "run-time". These TTPs are called monitoring agents. Even though in theory one can sign a contract without a TTP, this is rarely done. TTP observe the system and in case of violations induce the further sanctioning process playing the role of *observer*.

4.3 López y López et al. [32]

The work by López y López et al. has its roots in the SMART agent framework. This framework is structured similar to our sanctioning process idea although no specification

Table 1. Contrasting the Sanctioning Process

platform	**violation detection**	**sanctioning determination**	**sanctioning application**	**assimilation**
eInstitutions [19,27]	regimentation (institutional enforcement)	specified within norms	automatic by staff agents	no system learning in initial version, only as learning by targets
OperA [17]	trusted third parties (typically some form of institutional watchmen)	specified in contract	monitor agents (i.e. typically institutional watchmen)	learning by targets
López y López et al. [32]	defenders (institutional watchmen)	defined using "secondary norms" (interlocked with the norm being violated)	defenders act as executors and controllers	centered around learning by targets
$\mathcal{MOISE}^{Ins}$ [6]	primarily institutional enforcement	supervisor agents (acting as observer and judge)	supervisor agents act as executors	consideration of system adaption (triggered by the targets)
EMIL-I-A [2]	peer control (third, second party)	dynamic adaptation heuristic	peers (third and second party)	dynamic, both targets (utilitarian and cognitive level) as well as legislators
Cliffe et al. [11,9,10]	both regimentation and enforcement (all specifications) possible	typically specified in the norms at design-time, but can also be done by agents at run-time	in run-time version any agent can act as enforcer; central enforcement using the governor/normative monitor also possible	learning by targets up to the agent designer; system learning is currently being worked on [15]

of the phases is given. Moreover, a much more abstract (compared to our utilitarian) concept of agents is considered. In the work by López y López et al. agents follow norms to obtain their normative goals, without considering how these goals are fulfilled. More specifically, the authors make a distinction between primary and secondary level norms, and their relation with the concept of interlocking. When a primary norm is violated, it activates (as they are interlocked) a second level norm in charge of sanctioning.

The agents responsible for the sanctioning on this platform are the norm *defenders*. In our role classification, they act as *observers*, *executors* and *controllers*. It is not clear who plays the role of the *judge*, as the second level (sanctioning) norm already specifies the sanction and different situations with conflicting norms that might influence the calculation of this sanction are considered.

One very interesting point of the work by López y López et al. is the formalization of an autonomous normative reasoning process. This process is composed of three others that allow agents to decide whether to adopt a norm, whether to comply with the norm, and to update the goals of agents. This process is very important in open distributed systems as it allows adaptation and dynamicity against this rapidly changeable environments.

4.4 $\mathcal{MOISE}^{Ins}$ [6]

The $\mathcal{MOISE}^{Ins}$ framework follows a similar structure to the eI approach. It employs generic supervisor agents, aiming at controlling and enforcing the rights and duties of autonomous "domain" agents operating in a normative organization expressed with $\mathcal{MOISE}^{Ins}$. Whereas supervisor agents (who play the roles of *observers*, *judges* and *executor*) are dedicated to the control of the system at the normative and sanctioning level, the domain agents implement the functionalities of the application level. Hence, $\mathcal{MOISE}^{Ins}$ envisions agents to be forced to comply with norms (although they have the freedom of violation) because the violations are being detectable by the system. One aspect of special interest to the sanctioning process that the authors of the model emphasise is the normative learning and adaption by the society as a result of the sanctioning act. Thus, in $\mathcal{MOISE}^{Ins}$ agents may decide to adapt and change the organization in a bottom-up process if they feel that the current normative system is not suitable, installing a new normative pattern/structure. The phases of our process are also very well identified from the observation by the supervisor agents, passing through the calculation of the sanction (done during the design of the system), and the application of the sanction by the supervisor agent.

4.5 EMIL-I-A [2]

EMIL-I-A is an agent architecture able to process the normative cues in the social environment where agents interact. This architecture allow agents to self-organize and regulate their own norm-compliance within the system in a decentralized manner, without the necessity of a central authority. Agents' decision of whether to follow the norms is orchestrated by the internalization mechanism agents are endowed with, and by the

norms' salience, as the baton that orchestrates the intelligent norm compliance. Following the philosophy of decentralization, agents assume all the roles, working as *observers*, *judges* and *executors*. Because of this unification of roles and adaptability of the EMIL-I-A architecture, agents have a plastic notion of norms, making the terms of compliance and violation also dynamic and dependent on agents' perception of the environment. The authors present a running example of their architecture using a simulation framework that approaches the establishment the of cooperation norm.

The identification of a norm-violation is achieved by peers, reducing the system's costs invested in specific norm-controllers, as happens in eI. Once a violation is detected, in the initial version of the work [2], the sanction was specified by the policy maker/system designer. However in a latter version [39], the authors developed a heuristic that allowed agents to intelligently change the sanctions associated to norm-violations. The determination and the application is done by the same peer-agent, and the punished agent automatically receives the sanction. Finally, when an EMIL-I-A agent receives the different types of sanctions studied, they assume the effect of such sanctions to follow empirical trends obtained from experiments with human subjects.

4.6 Cliffe et al. [9,10]

The Inst*AL* framework by Cliffe et al. [11,12,9,10] is a normative framework architecture with a formal mathematical model to specify, verify and reason about the norms that might be operational in an open distributed system. The Inst*AL* approach has opted for an event-driven normative approach: violations are based on the events/actions rather than states and the norms are centrally located and not explicitly internalised within the system's participants. The premise of the model is that events trigger the creation of normative fluents. Inspired by Jones and Sergot's [30] account of institutional power and the notion of 'counts-as', the generation relation is used to explain the connection between actions and their interpretation in the context of the normative framework. The Inst*AL* framework enables to reason about normative systems and verify their properties (e.g. whether certain states be reached, and which (sequence of) actions result in which states) at design-time. The Inst*AL* framework can thus be used to reason about violation actions, however what is important to note is that it can only reason about violation actions that have been defined as violations in the design-time specifications. Emergent properties and changes in the normative setting are currently not considered, but future extensions taking into account these properties are being worked on (see [15] for example).

With the perspective of run-time normative systems, a methodology to to migrate Inst*AL*'s design-time (verification) set of norms to a run-time set presented in [4]. Norm compliance within a running Inst*AL* framework is enabled by a Governor/normative monitor entity who monitors the events in the system that are relevant to the normative system. The normative monitor can be configured to observe all actions in the system, as well as be configured as passive logging entity, that does not monitor directly itself but only reacts to messages from agents that observe a violation. These agents could be the following: second party and third party agents, normative-empowered agents as well as the violating agents themselves. The set-up is up to the system designer. The same is true for the sanctioning calculation and enforcement action. Thus, depending

on the context, the normative monitor can act as the observer, sanction calculator and/or enforcer or can leave it up the the participants to use its logged violation information how they see fit. The normative monitor can also act as a normative information base for the participant to use normative planning purposes. The Inst*AL* framework thus leaves it to the system designer to set up the sanctioning as required. It can incorporate any of the settings described in the first three stages of our process. Concerning the final stage – the assimilation, some work has been done by Corapi et al. [15] to allow for an inductive learning about the norms of the system from a system designer's point of view. Again the learning on the side of the agents can be incorporated by the agent designer as required and is not specifically regimented by the Inst*AL* framework.

5 Closing Remarks

This paper is a toddling start towards an organization of features akin to sanctioning, to facilitate their use in the design of regulated open MAS. We chose the unconventional perspective of sanctioning as a process, in order to plot a field where available resources are easily harvested while unexplored opportunities are more crisply revealed. Thus, we outlined the bounds of the punitive process and advanced core conceptual distinctions. Our quick exploration of these elements in the context of five MAS platforms suggests our proposal is valid and points out obvious topics to address next. In this last section we will touch upon the not so obvious.

We have sidestepped the fundamental question of why sanctioning works. If addressed from the assumption that sanctioning not only motivates but *modifies* behavior, it entails, first, an examination of tenets about an agent's rationality; in particular, in relation to the agent's motivations and in the context of the agent's conformity to a society. It also entails an analysis of the type of modification that is desired or achievable through sanctioning, those factors that bear upon behavior modification and the way behavior modification is achieved and ascertained. Utilitarianism on one hand, and on the other BDI architectures of different flavors, are standard approaches to avoid many of the problematic aspects of rationality. With respect to punishment, however, even these simplifications still leave hard questions open. At any rate, our description of the four-stage process should have made clear that a designer of sanctioning mechanisms for a given MAS may have to consider many different facets of the mechanism, even assuming absolute control over the cognitive capabilities of agents.

A similar type of concern, likely as rich, may arise with respect to the "assimilating sanctioning" stage of the punitive process. We hinted at two foci of analysis: individuals and system, hinted at the pertinence of the abundant work on agent and MAS evolution and, in particular, that of emergence and immersion of norms, but there is more. The process of sanctioning may be understood, also, as a resource to modulate those dynamics thus making evolution faster, or more sensitive to the success of sanctioning, or the other way around to take the intended evolution as the modulator of sanctioning. In both directions, the study of how to determine the efficiency of a normative system and its sensitivity to changes in sanctioning is paramount. Of special significance is the interplay of these matters with the moral and cognitive disposition of agents.

Our last comment is on implementational aspects of the punitive process. The distinctions we outlined in this paper are but a fraction of the number of design features

that are amenable for use in actual MAS. Still more work is needed to come up with a concise framework for a design methodology and the development and assembly of tools that allow for a convenient use of a sanctioning process on top of generic MAS development platforms. The first steps should be to start with platforms like the five we examined and make sanctioning available as a service on top.

Acknowledgments. This work was supported by the Spanish Education and Science Ministry [Engineering Self-* Virtually-Embedded Systems (EVE) project, TIN2009-14702-C02-01]; Proyecto Intramural de Frontera MacNorms [PIFCOO-08-00017] and the Generalitat de Catalunya [2009SGR1434]. Daniel Villatoro is supported by a CSIC predoctoral fellowship under JAE program. Tina Balke was partially supported by a scholarship of the German Academic Exchange Service as well as by funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 288147. Both authors thank Pablo Noriega for his valuable discussion of and comments on the paper.

References

1. Ågotnes, T., van der Hoek, W., Tennenholtz, M., Wooldridge, M.: Power in normative systems. In: AAMAS 2009: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 145–152. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2009)
2. Andrighetto, G., Villatoro, D., Conte, R.: Norm internalization in artificial societies. AI Communications 23, 325–339 (2010)
3. Balke, T.: A taxonomy for ensuring institutional compliance in utility computing. In: Boella, G., Noriega, P., Pigozzi, G., Verhagen, H. (eds.) Normative Multi-Agent Systems. Dagstuhl Seminar Proceedings, vol. 09121. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2009)
4. Balke, T., De Vos, M., Padget, J., Traskas, D.: Normative run-time reasoning for institutionally-situated bdi agents. In: Proceedings - 2011 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2011. IEEE Computer Society (2011)
5. Becker, G.S.: Crime and punishment: An economic approach. The Journal of Political Economy 76(2), 169–217 (1968)
6. Boissier, O., Gâteau, B.: Normative multi-agent organizations: Modeling, support and control, draft version. In: Boella, G., van der Torre, L., Verhagen, H. (eds.) Normative Multi-agent Systems. Dagstuhl Seminar Proceedings, vol. 07122. Internationales Begegnungs– und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
7. Bradshaw, J.M., Dutfield, S., Carpenter, B., Jeffers, R., Robinson, T.: KAoS: A generic agent architecture for aerospace applications. In: Proceedings of the CIKM 1995 Workshop on Intelligent Information Agents (1995)
8. Carpenter, J.P.: Punishing free-riders: How group size affects mutual monitoring and the provision of public goods. Games and Economic Behavior 60(1), 31–51 (2007)
9. Cliffe, O.: Specifying and Analysing Institutions in Multi-agent Systems Using Answer Set Programming. PhD thesis, University of Bath (2007)
10. Cliffe, O., De Vos, M., Padget, J.: Modelling Normative Frameworks Using Answer Set Programing. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 548–553. Springer, Heidelberg (2009)

11. Cliffe, O., De Vos, M., Padget, J.: Specifying and Analysing Agent-Based Social Institutions Using Answer Set Programming. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 99–113. Springer, Heidelberg (2006)
12. Cliffe, O., De Vos, M., Padget, J.: Specifying and Reasoning About Multiple Institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006 Workshops. LNCS (LNAI), vol. 4386, pp. 67–85. Springer, Heidelberg (2007)
13. Coleman, J.S.: Foundations of Social Theory. Belknap Press (August 1998)
14. Conte, R., Andrighetto, G., Campenni, M.: Internalizing norms. A cognitive model of (social) norms' internalization. The International Journal of Agent Technologies and Systems (IJATS) 2(1), 63–73 (2010)
15. Corapi, D., De Vos, M., Padget, J., Russo, A., Satoh, K.: Normative design using inductive learning. Theory and Practice of Logic Programming 11, 783–799 (2011)
16. Dibbell, J.: Mutilated furries, flying phalluses: Put the blame on griefers, the sociopaths of the virtual world. WIRED Magazine (16-02) (2008)
17. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. PhD thesis, Utrecht University (2003)
18. Dreber, A., Rand, D., Fudenberg, D., Nowak, M.: Winners don't punish. Nature 452, 348–351 (2008)
19. Esteva, M.: Electronic Institutions: from specification to development. PhD thesis, Artificial Intelligence Research Institute, IIIA (2003)
20. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an electronic institutions editor. In: AAMAS 2002: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1045–1052. ACM, New York (2002)
21. Esteva, M., Padget, J., Sierra, C.: Formalizing a Language for Institutions and Norms. In: Meyer, J.-J.C., Tambe, M. (eds.) Intelligent Agents VIII. LNCS (LNAI), vol. 2333, pp. 348–366. Springer, Heidelberg (2002)
22. Esteva, M., Rodríguez-Aguilar, J.A., Arcos, J.L., Sierra, C., García, P.: Formalizing agent mediated electronic institutions. In: Proceedings of the Congrès Català d'Intel.ligència Artificial, pp. 329–338 (2000)
23. Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., Arcos, J.-L.: On the Formal Specification of Electronic Institutions. In: Dignum, F., Sierra, C. (eds.) Agent Mediated Elec. Commerce. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
24. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Jennings, N.R., Sierra, C., Sonenberg, L., Tambe, M. (eds.) Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), vol. 1, pp. 236–243. IEEE Computer Society, Washington, DC (2004)
25. Fehr, E., Gächter, S.: Cooperation and punishment in public goods experiments. The American Economic Review 90(4), 980–994 (2000)
26. Fishburn, P.C.: Utility theory for decision making. Publications in Operations Research (18) (1970)
27. García-Camino, A.: Normative regulation of open multi-agent systems. PhD thesis, Artificial Intelligence Research Institute (IIIA), Spain (2009)
28. Grossi, D., Aldewereld, H., Dignum, F.: *Ubi Lex, Ibi Poena*: Designing Norm Enforcement in E-Institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006 Workshops. LNCS (LNAI), vol. 4386, pp. 101–114. Springer, Heidelberg (2007)
29. Gurerk, O., Irlenbusch, B., Rockenbach, B.: The competitive advantage of sanctioning institutions. Science 312(5770), 108–111 (2006)

30. Jones, A.J., Sergot, M.: A Formal Characterisation of Institutionalised Power. ACM Computing Surveys 28(4), 121 (1996)
31. López y López, F., Luck, M.: Modelling norms for autonomous agents. In: Chavez, E., Favela, J., Mejia, M., Oliart, A. (eds.) Fourth Mexican International Conference on Computer Science, pp. 238–245. IEEE Computer Society (2003)
32. López y López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. In: Boella, G., van der Torre, L., Verhagen, H. (eds.) Normative Multi-agent Systems. Dagstuhl Seminar Proceedings, vol. 07122. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
33. Minsky, N.H.: Law-governed systems. Software Engineering Journal - Special Issue on Software Process and its Support 6(5), 285–302 (1991)
34. Perreau De Pinninck, A., Sierra, C., Schorlemmer, M.: A multiagent network for peer norm enforcement. Autonomous Agents and Multi-Agent Systems 21, 397–424 (2010)
35. Perreau de Pinninck Bas, A.: Techniques for Peer Enforcement in Multiagent Networks. PhD thesis, Universitat Autónoma de Barcelona, Spain (2010)
36. Scheve, C., Moldt, D., Fix, J., Luede, R.: My agents love to conform: Norms and emotion in the micro-macro link. Journal of Computational & Mathematical Organization Theory 12(2-3), 81–100 (2006)
37. Staller, A., Petta, P.: Introducing emotions into the computational study of social norms: A first evaluation. Journal of Artificial Societies and Social Simulation 4(1) (2001)
38. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Norms in multiagent systems: from theory to practice. International Journal of Computer Systems Science & Engineering 20(4), 95–114 (2004)
39. Villatoro, D., Andrighetto, G., Sabater-Mir, J., Conte, R.: Dynamic sanctioning for robust and cost-efficient norm compliance. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence/Association for the Advancement of Artificial Intelligence, pp. 414–419 (2011)
40. Villatoro, D., Sen, S., Sabater-Mir, J.: Of social norms and sanctioning: A game theoretical overview. International Journal of Agent Technologies and Systems 2, 1–15 (2010)

Overcoming Omniscience for Norm Emergence in Axelrod's Metanorm Model

Samhar Mahmoud[1], Nathan Griffiths[2], Jeroen Keppens[1], and Michael Luck[1]

[1] Department of Informatics
King's College London
London WC2R 2LS, UK
samhar.mahmoud@kcl.ac.uk
[2] Department of Computer Science
University of Warwick
Coventry CV4 7AL, UK

Abstract. Norms are a valuable mechanism for establishing coherent cooperative behaviour in decentralised systems in which no central authority exists. In this context, Axelrod's seminal model of norm establishment in populations of self-interested individuals [2] is important in providing insight into the mechanisms needed to support this. However, Axelrod's model suffers from significant limitations: it adopts an evolutionary approach, and assumes that information is available to all agents in the system. In particular, the model assumes that the private strategies of individuals are available to others, and that agents are omniscient in being aware of all norm violations and punishments. Because this is an unreasonable expectation, the approach does not lend itself to modelling real-world systems such as peer-to-peer networks. In response, this paper proposes alternatives to Axelrod's model, by replacing the evolutionary approach, enabling agents to learn, and by restricting the metapunishment of agents to only those where the original defection is perceived, in order to be able to apply the model to real-world domains.

1 Introduction

In many application domains, engineers of distributed systems may choose, or be required, to adopt an architecture in which there is no central authority and the overall system consists solely of self-interested autonomous agents. The rationale for doing so can range from efficiency reasons to privacy requirements. In order for such systems to achieve their objectives, it may nevertheless be necessary for the behaviour of the constituent agents to adhere to certain constraints, or *norms*. In peer-to-peer file sharing networks, for example, we require (at least a proportion of) peers to provide files in response to the requests of others, while in wireless sensor networks nodes must share information with others for the system to determine global properties of the environment. However, there is typically a temptation in such settings for individuals to deviate from the desired behaviour. For example, to save bandwidth peers may not provide files, and to

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 186–202, 2012.

conserve energy the nodes in a sensor network may not share information. It is therefore desirable to minimise the temptation for agents to deviate from the desired behaviour, and encourage the emergence of cooperative norms.

Norms have been studied by very many different researchers, over several different areas (for example, [6–8, 16, 18–20, 23]. Most notably, Axelrod's seminal investigation of norm establishment in populations of self-interested individuals [2] provides an analysis of the conditions in which norms can be established. In his experiments, a population of agents repeatedly play a simple game, in which agents make decisions about whether to comply with a desired norm of cooperation and whether to punish those who are seen to violate this norm. These decisions may result in certain penalties or rewards, with the strategies of agents being determined through an evolutionary process, in which the more successful strategies are reproduced. In this setting, Axelrod explored how the emergence of norm compliant strategies can be encouraged.

Although Axelrod's investigation is successful in establishing cooperative norms, the model makes several assumptions that are unrealistic in real-world settings. In particular, in many domains it is not possible to remove unsuccessful agents and replicate those that are more successful, and there is no centralised control that could oversee this process. Instead, we need a mechanism through which individuals can learn to improve their strategies over time. If we enable individuals to compare themselves to others, and adopt more successful strategies, then we can take a *learning interpretation* of the evolutionary mechanism [13], without needing to remove and replicate individuals. However, this learning interpretation requires that the private strategies of individuals are available for observation by other agents, which is again an unreasonable assumption. Furthermore, as has been shown elsewhere, Axelrod's model is unable to sustain cooperation over a large number of generations [10]. Axelrod's approach, as discussed below, relies on agents being able to punish both those that defect and those that fail to punish defection, yet this is unrealistic since it assumes *omniscience* through agents being aware of all norm violations and punishments.

In this paper we investigate alternatives that allow us to make use of the mechanisms resulting from Axelrod's investigations, in more realistic settings. Specifically, we first take a learning interpretation of evolution and describe an alternative technique, strategy copying, which prevents norm collapse in the long term. Second, we remove the assumption of omniscience and constrain the ability of agents to punish according to the defections they have observed. Finally, to obviate the need for information on the private strategies of others, we propose a learning algorithm through which individuals improve their strategies based on their experience.

The paper begins by reviewing Axelrod's original norms game and metanorms game, in which our work is situated. Then, in Section 3, we present our strategy copying technique, and show how it performs in the original context and in situations in which observation of defection is not guaranteed. In Section 4, we describe a reinforcement learning algorithm designed to avoid the need for

access to the private strategies of others. Section 5, considers related work, before presenting our conclusions in Section 6, with a discussion of the significance of our results.

2 Axelrod's Model

2.1 The Norms Game

Axelrod's *norms game* adopts an evolutionary approach in which successful strategies multiply over generations, potentially leading to convergence on cooperative norms [2]. Each agent in the population has a number of opportunities (o) in which it can choose to *defect* by violating a norm, and such behaviour has a particular known probability of being observed, or *seen* (S_o). An agent i has two decisions, or strategy dimensions, as follows. First, it must decide whether to defect, determined by its *boldness* (B_i); and second, if it sees another agent defect in a particular opportunity (with probability S_o) it must decide whether to punish this defecting agent, determined by its *vengefulness* (V_i), which is the probability of doing so. If $S_o < B_i$ then i defects, receiving a *temptation payoff*, $T = 3$, while *hurting* all other agents with payoff $H = -1$. If a defector is *punished* (P), it receives an additional punishment payoff of $P = -9$, while the punishing agent pays an *enforcement cost*, $E = -2$. The initial values of B_i and V_i are chosen at random from a uniform distribution of a range of 8 values between $\frac{0}{7}$ and $\frac{7}{7}$.

Axelrod's simulation had 20 agents, with each having four opportunities to defect, and the chance of being seen for each drawn from a uniform distribution between 0 and 1. After playing a full round (all four opportunities), scores for each agent are calculated to produce a new generation, as follows. Agents that score better or equal to the average population score plus one standard deviation are reproduced twice in the new generation. Agents that score one standard deviation or more under the average score are not reproduced, and all others are reproduced once. Finally, a mutation operator is used to enable new strategies to arise. Since B_i and V_i (which determine agent behaviour) take eight possible values they can be represented by three bits, to which mutation is applied (by flipping a bit) when an agent is reproduced, with a 1% chance.

In this model, cooperative norms are established when V_i is high and B_i is low for all members of the population, so that defection is unlikely, and observed defections are likely to be punished. In 100 generations, Axelrod found only partial establishment of a norm against defection, so introduced an additional mechanism to support norms in his *metanorm* model.

2.2 The Metanorms Game

The key idea underlying Axelrod's metanorm mechanism is that some further encouragement for enforcing a norm is needed. In the *metanorms* game, if an

agent sees a defection but does not punish it, this is itself considered as a form of defection, and others in turn may observe this defection (with probability S_o) and apply a punishment to the non-enforcing agent. As before, the decision to punish is based on vengefulness, and brings the defector (namely, the non-punisher) a punishment cost of $P' = -9$ and the punisher an enforcement cost of $E' = -2$. Applying the simulation to the metanorms game gives runs with high vengefulness and low boldness, which is exactly the kind of behaviour needed to support the establishment of a norm against defection.

However, Axelrod's analysis of results was limited. As has been shown subsequently, allowing Axelrod's *metanorms game* to run for an extended period (1,000,000 generations) ultimately results in norm collapse [9]. As Mahmoud et al. have shown [10], this norm collapse arises as a consequence of two aspects. First, a sufficiently long run (compared to Axelrod's limited run of 100 generations) provides the opportunity for a sequence of mutations to cause norm collapse even after a norm has been established in the population. Second, such mutation is magnified by the evolutionary manner of replication, generating a new population of agents.

3 Strategy Copying

As indicated above, the evolutionary approach causes some problems in extended runs, leading to norm collapse. In addition, for use in domains such as peer-to-peer or wireless sensor networks, the agents themselves cannot be deleted or replicated, but instead must modify their own behaviour. In this section, therefore, we examine a simple alternative to Axelrod's model in which an agent that performs poorly in comparison to others in the population can *learn* new strategies (in terms of vengefulness and boldness attributes) by adopting the strategy of other, better performing agents, replacing the existing strategy with a new one. Agents can achieve this in different ways: they can copy the strategy of the agent with the highest score or they can copy the strategy of one of the group of agents that perform best in the population.

3.1 Strategy Copying from a Single Agent

Intuitively, copying the strategy of the agent with the highest score appears to be a promising approach. However, it leads to poor results in the long term because it draws strategies from only one agent rather than a population of agents. This makes the approach vulnerable to strategies that are only successful in a small number of possible settings. Moreover, by failing to draw strategies from a variety of agents, the strategies tend to converge prematurely. To illustrate, consider a group of students taking an examination, with one of the students having cheated. If the cheating student has not been seen, they may achieve the best exam performance. However, if all other students copy this behaviour and cheat in the next exam, there is a high possibility that they will be caught, and will thus

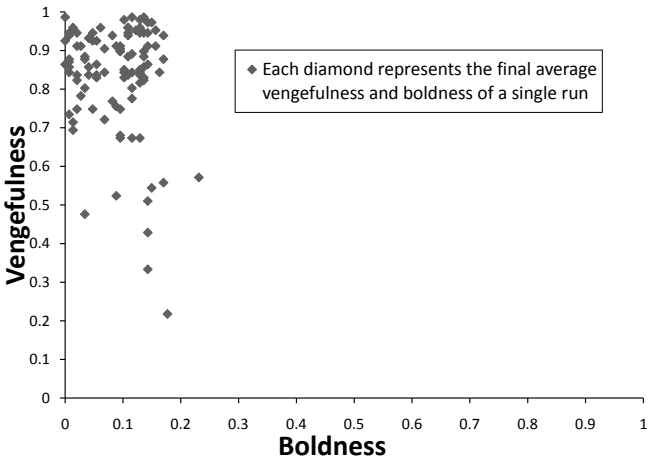


Fig. 1. Strategy copying from the best agent; 100 timesteps

suffer from much worse results than if they had not cheated. This is supported by the results shown in Figures 1 and 2, illustrating experiments with runs of 100 and 1,000,000 timesteps (where a timestep represents one *round* of agents having opportunities to defect and learning from the results, and is equivalent to a *generation* in the evolutionary approach). Each point on the graph (shown as a diamond to increase visibility) represents the average boldness and vengefulness of the population at the end of a single simulation run.

In the short term, as can be seen from Figure 1, copying from the best agent leads to norm establishment. However, in the long term the norm collapses, as shown in Figure 2. This can be explained by the fact that an agent with low vengefulness that does not punish a defector (and thus does not pay an enforcement cost) but is also not metapunished, scores better than any other agent with high vengefulness that does punish (and thus pays the enforcement cost). As a result, other agents copy the low vengefulness of this agent so that low vengefulness becomes prevalent in the population. In the same way, when low vengefulness prevails in the population, an agent with high boldness defects, gaining a *temptation payoff*, and hurting others without receiving punishment. As a result, other agents copy the high boldness of this agent so that low vengefulness and high boldness are propagated through the population, leading to norm collapse. This transition from high vengefulness to low vengefulness and from low boldness to high boldness requires time to manifest, but the duration of the period of time is not fixed.

3.2 Strategy Copying from a Group of Agents

Alternatively, and as we have suggested, we might seek to copy the strategy of one in a group of high-performing agents. In this view, agents choose one

Fig. 2. Strategy copying from the best agent; 1,000,000 timesteps

agent, at random, from the group of agents with scores above the average, and copy its strategy. As previously, experiments of different durations (between 100 and 1,000,000 timesteps) were carried out; the results in Figure 3, for 1,000,000 timesteps, show that all runs ended with norm establishment in the long term, indicating that this approach is effective in eliminating the problematic effect of the replication method. This approach avoids norm collapse since it does not limit itself to the best performing agent, and thus does not run the risk of only adopting a strategy that performs well in a small number of settings.

3.3 Observation of Defection

As stated in Section 2, in Axelrod's model, an agent Z is able to punish another agent Y that does not punish a defector X, even though agent Z did not see the defection of agent X. However, such metapunishment is not possible if the original defection is not observed: guaranteed observation of the original defection is an unreasonable expectation in real-world settings. In consequence, our model needs adjustment so that metapunishment is only permitted if an agent observes the original defection. However, because this observation constraint limits the circumstances in which metapunishment is possible, its introduction corresponds to removing the metapunishment component from part of the game. In Axelrod's original experiments, metapunishment was introduced as a means to stabilise an established norm. In his setting, norms tend to collapse shortly after they are established without metapunishment. In fact, this remains the case in our model and our results confirm this.

More precisely, the observation constraint causes all runs to end in norm collapse when simulations are run for 1,000,000 timesteps, as shown in Figure 4.

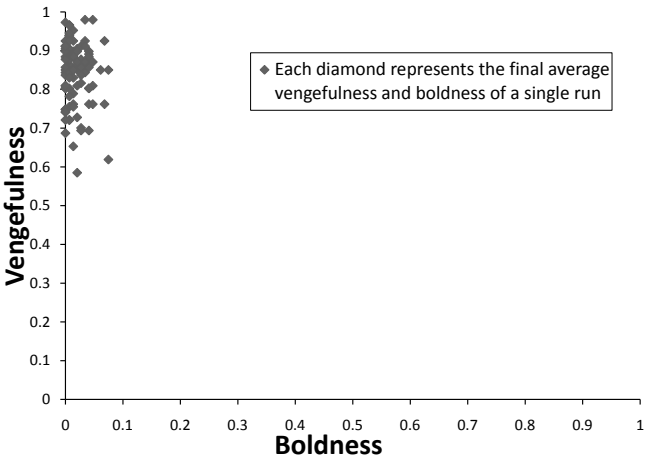


Fig. 3. Strategy copying from a group of agents

This is due to the fact that, as in the original model, runs initially stabilise on high vengefulness and low boldness, and then mutation causes vengefulness to reduce. If an agent Y with high vengefulness and low boldness changes through mutation to give lower vengefulness, while boldness for all remains low, there is no defection and the mutated agent survives. In addition, if boldness then mutates to become just a little higher for a different agent X, with average vengefulness remaining high, X will still rarely defect because of relatively low boldness.

If it *does* defect, however, and *is* seen by others, it receives a low score, unless it is *not* punished, in which case the non-punishing agents may themselves be punished because of the high vengefulness in the general population. Here, agent Y may not punish X because of the low probability of being seen (which must be below the low boldness level to have caused a defection) or because it has mutated to have lower vengefulness. In the former case, Y will not be metapunished for non-punishment (since there is a low probability of some other agent Z having seen it), but in the latter case, Y might be metapunished if it is seen by others. The likelihood of agent Y's non-punishment being seen requires first X's defection being seen by Y, and then Y's non-punishment being seen by others. Importantly, in this new model, agents that metapunish Y must themselves see X's defection. Since this combination of requirements is rare, such mutants survive for a longer duration, enabling their strategy to propagate through the population, and causing vengefulness to decrease. In addition, if another such event occurs, it will cause vengefulness to drop further until it reaches a very low level. When the model runs over an extended period, such a sequence of events is much more likely, and low vengefulness allows a mutant of higher boldness to survive and spread among the whole population, which is the cause of the norm collapse.

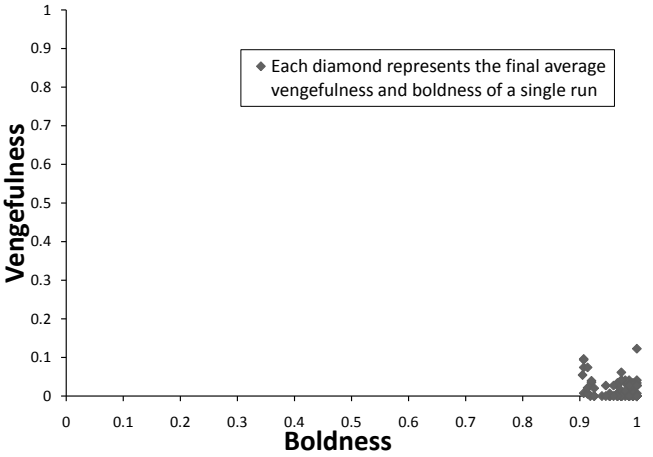


Fig. 4. Strategy copying with observation of defection

4 Strategy Improvement

Once the observation constraint is introduced, strategy copying becomes inadequate. Furthermore, it requires that agents have access to the strategies and decision outcomes of others in order to enable the copying mechanism. As we have argued, in real-world settings such observations tend to be unrealistic. *Reinforcement learning* offers an alternative to Axelrod's evolutionary approach to improving performance of the society while keeping agent strategies and decision outcomes private. There are many reinforcement techniques in the literature, such as Q-learning [21], PHC and WOLF-PHC [4], which we use as inspiration in developing a learning algorithm for strategy improvement in the metanorms game.

4.1 Q-Learning

Q-learning is a reinforcement learning technique that allows the learner to use the (positive or negative) reward, gained from taking a certain action in a certain state, in deciding which action to take in the future in the same state. Here, the learner keeps track of a table of Q-values that record an action's quality in a particular state, and updates the corresponding Q-value for that state after each action. The new value is a function of the old Q-value, the reward received, and a learning rate, δ, and the action with the highest updated Q-value for the current state is chosen. However, for us, Q-learning suffers from two drawbacks. First, it considers an agent's past decisions and corresponding rewards, which are not relevant here; doing so would inhibit an agent's ability to adapt to new circumstances. Second, actions are precisely determined by the Q-value; there is no probability of action, unlike Axelrod's model.

Bowling and Veloso [4] proposed policy hill climbing (PHC), an extension of Q-learning that addresses this latter limitation. In PHC, each action has a probability of execution in a certain state, determining whether to take the action. Here, the probability of the action with the highest Q-value is increased according to a learning rate δ, while the probabilities of all other actions are decreased in a way that maintains the probability distribution, with each probability update occurring immediately after the action. In enhancing the algorithm, a *variable* learning rate is introduced, which changes according to whether the learner is winning or losing, inspired by the WOLF technique (win or learn fast). This suggests two possible values for δ: a low one to be used while an agent is performing well and a high one to be used while the agent is performing poorly.

However, in one round of Axelrod's game, an agent can perform multiple punishments (potentially one per defection and non-punishment observed), while only having a small number of opportunities to defect (four in Axelrod's configuration). Therefore, punishment and metapunishment actions would be considered much more frequently than defection, leading to disproportionate update of probabilities of actions, with some converging more quickly than others. To address this imbalance, we can restrict learning updates to occur only at the end of each round, rather than after each individual action, so that boldness and vengefulness are reconsidered once in each round and evolve at the same speed. The aim here is to change the probability of action significantly when losing, while changing it much less when winning, providing more opportunities to adapt to good performance.

While basic Q-learning is not appropriate because of the lack of a probability of taking action, PHC-WOLF suffers from a disproportionate update of probabilities of action. Nevertheless, the use of the variable learning rate approach in PHC-WOLF is valuable in providing a means of updating the boldness and vengefulness values in determining which action to take. However, since agents that perform well need not change strategy, we can consider only one learning rate. The next section details our algorithm, inspired by this approach.

4.2 BV Learning

To address the concerns raised above, in this section, we introduce our BV learning algorithm. This requires an understanding of the relevant agent actions and their effect on boldness and vengefulness, as summarised in Table 1, which outlines the different actions available to an agent and the consequences of each on the agent's score.

Now, since boldness is responsible for defecting, an agent that obtains a good score as a result of defecting should increase its boldness, and an agent that finds defection detrimental to its performance should decrease its boldness. Learning suitable values for vengefulness is more complicated, since while it is responsible for both punishment and metapunishment, these also cause enforcement costs that decrease an agent's score. Low vengefulness allows an agent to avoid paying an enforcement cost, but can result in receiving metapunishment. Vengefulness thus requires a consideration of all these aspects. This intuition is formalised as

Table 1. Effects of decisions on score

Decision	**Effects**
Defect	Gain temptation payoff Hurts all other agents Potentially suffer punishment cost
Cooperate	—
Punish	Punisher pays enforcement cost Defector pays punishment cost
Not punish	Potentially suffer metapunishment (incurring punishment cost)
Metapunish	Punisher pays enforcement cost Defector pays punishment cost
Not metapunish	—

in Algorithm 1, as follows. (Note that we use subscripts to indicate the relevant agent only when needed.)

First, in order to determine the unique effect of each individual action on agent performance, note that we decompose the single combined total score (TS) of the original model into distinct components, each reflecting the effect of different classes of actions. The defection-cooperation action brings about a change only if an agent defects (Line 9): the agent's score increases by a *temptation payoff*, T (Line 10), but it *hurts* all others in the population, whose scores decrease by H (line 12), where H is a negative number that is thus added to the score. If an agent cooperates, no scores change. We can therefore use just one distinct value to keep track of this score, referred to as the *defection score* (DS), and which determines whether to increase or decrease boldness.

Conversely, punishment and metapunishment both have two-sided consequences: if an agent j sees agent i defect in one of its opportunities (o) to do so, with probability S_o (Line 13), and decides to punish it (which it does with probability V_j; Line 14), i incurs a punishment cost, P, to its DS (Line 15), while the punishing agent incurs an enforcement cost, E, to a different score, its *punishment score*, PS (Line 16). Note that both P and E are negative values, so they are added to the total when determining an overall value. As the name suggests, PS captures the total score obtained by an agent as a result of punishing another, and applies to both punishment and metapunishment (enforcement costs). There is also a different change (resulting from potential subsequent received metapunishment) if it decides not to punish (Line 17). If j does not punish i, and another agent k sees this in the same way as previously (Line 19), and decides to metapunish (Line 20), then k incurs an enforcement cost, E, to its PS, and j incurs a punishment cost P to its *no punishment score*, NPS. (An agent's NPS is obtained from not punishing, and comprises the metapunishment cost alone.)

In Axelrod's original model, those agents that are one standard deviation or more below the mean are eliminated and replaced in the subsequent population

Algorithm 1. The Simulation Control Loop: *simulation*$(T, H, P, E, \gamma, \delta)$

for each agent i **do**
  {***Initialising***}
  $B_i = random()$ {**Random generator that uses uniform distribution**}
  $V_i = random()$ {**Random generator that uses uniform distribution**}
for each round **do**
  for each agent i **do**
    {***Decision making***}
    for each opportunity to defect o **do**
      if $B_i > S_o$ **then**
        $DS_i = DS_i + T$
        for each agent j: $j \neq i$ **do**
          $TS_j = TS_j + H$
          if see(j,i,S_o) **then**
            if punish (j, i, V_j) **then**
              $DS_i = DS_i + P$
              $PS_j = PS_j + E$
            else
              for each agent $k : k \neq i \wedge k \neq j$ **do**
                if see(k,j,S_o) **then**
                  if punish (k, j, V_k) **then**
                    $PS_k = PS_k + E$
                    $NPS_j = NPS_j + P$
  $Temp = 0$
  for each agent i **do**
    $TS_i = TS_i + DS_i + PS_i + NPS_i$
    $Temp = Temp + TS_i$
  $AvgS = Temp/no\_agents$
  for each agent i **do**
    {***Learning***}
    if $TS_i < AvgS$ **then** {$AvgS$ **is the mean score of all agents**}
      if explore(γ) **then**
        $B_i = random()$
        $V_i = random()$
        if $DS_i < 0$ **then**
          $B_i = max(B_i - \delta, 0)$
        else
          $B_i = min(B_i + \delta, 1)$
        if $PS_i < NPS_i$ **then**
          $V_i = max(V_i - \delta, 0)$
        else
          $V_i = min(V_i + \delta, 1)$

generation with new agents following the strategy captured by the boldness and vengefulness values of those agents that are one standard deviation or more above the mean. Thus, poorly performing agents are replaced by those that perform much better. In contrast, in our model, we distinguish more simply between good and poor performance, with only agents that score below the mean reconsidering their strategy. Thus, for each agent, we combine the various component scores into a total, TS and, if the agent is performing poorly (in relation to the average score, $AvgS$ in Line 30), we reconsider its boldness and vengefulness. Note that this average score is established through the lines in the algorithm around 27.

Now, in order to ensure we allow a degree of exploration (similar to mutation in the original model's evolutionary approach, to provide comparability) and to enable an agent to step out of the learning trend, here we adopt an *exploration rate*, γ, which regulates adoption of random strategies from the available strategies universe (Line 31). If the agent does not explore then, if defection is the cause of a low score (Line 34), an agent decreases its boldness, and increases it otherwise. Similarly, agents increase their vengefulness if they find that the effect of not punishing is worse than the effect of punishing (Line 38), and decrease vengefulness if the situation is reversed. As both PS and NPS represent the result of two mutually exclusive actions, their difference for a particular agent determines the change to be applied to vengefulness. For example, if $PS > NPS$, then punishment has some value, and vengefulness should be increased.

Finally, given a decision on whether to modify an agent's strategy, the degree of the change, or *learning rate* (δ), must also be considered. Since vengefulness and boldness have eight possible values from $\frac{0}{7}$ to $\frac{7}{7}$, we adopt the conservative approach of increasing or decreasing by one level at each point, corresponding to a learning rate of $\delta = \frac{1}{7}$. Thus, an agent with boldness of $\frac{5}{7}$ and vengefulness of $\frac{3}{7}$ that decides to defect less and punish more will decrease its boldness to $\frac{4}{7}$ and increase its vengefulness to $\frac{4}{7}$.

4.3 Evaluation

The algorithm is designed to mimic the behaviour of Axelrod's evolutionary approach as much as possible, while relaxing Axelrod's unrealistic assumptions. This allows us to replicate Axelrod's results and investigate his approach in more realistic problem domains. The analysis of a sample run reveals that agents with low vengefulness and agents with high boldness start changing their strategies. Here, agents with high boldness defect frequently, and are punished as a result, leading to a very low DS, in turn causing these agents to decrease their boldness. Agents with low vengefulness do not punish and are consequently frequently metapunished; as a result, their PS is much better (lower in magnitude) than their NPS, causing them to increase their vengefulness. The population eventually converges to comprise only agents with high vengefulness and low boldness. While noise is still introduced via the exploration rate causing random strategy adoption, the learning capability enables agents with such random strategies to adapt quickly to the trend of the population.

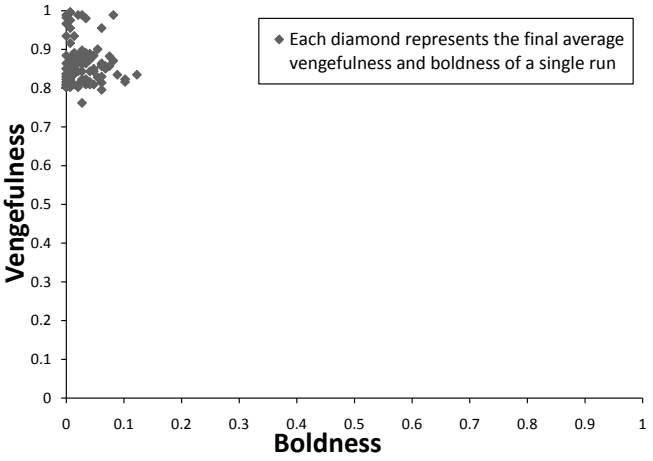


Fig. 5. Strategy improvement (with $\gamma = 0.01$)

As before, we also consider the problem of ensuring that an original defection is observed in order to provide a metapunishment. Introducing this constraint into our new algorithm, we ran experiments over different periods, with results indicating that norm establishment is robust in all runs. An example run for 1,000,000 timesteps is shown in Figure 6. This is because agents that use this new learning algorithm only change their strategy incrementally without wholesale change at any single point. The effect of a mutant with low vengefulness is not significant since, while the mutant might survive for a short period and cause some agents to change their vengefulness, any such change will be slight. It thus does not prevent such agents from detecting the mutant subsequently, in turn causing the mutant to increase its vengefulness.

5 Related Work

In multi-agent systems, research on norm propagation can be divided into two distinct approaches: top-down and bottom-up. In the *top-down* approach, a norm is introduced through a certain *authority*, which is then responsible for the monitoring and enforcement of this norm. In the *bottom-up* approach, agents discover and learn about the norm as a direct result of their interactions and, in most cases, there is no central authority that can enforce such norms. The former approach has been studied and analysed by many (for example, [22, 1, 3]), and this is not the focus of the work in this paper. In contrast, the bottom-up approach, also known as *norm emergence*, has not received the same sort of attention and this is just what this paper addresses.

Nevertheless, the basic notions underlying norm emergence, as understood in this paper, have themselves also been recognised and considered previously. For example, like the work in this paper, Epstein [7] also used imitation techniques in

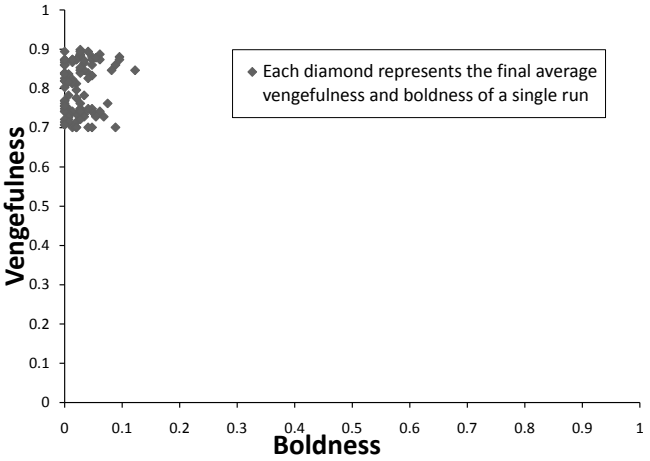


Fig. 6. Strategy improvement with defection observation (with $\gamma = 0.01$)

the context of norm emergence. In his model, agents must decide which side of a road to drive on, where the decision of each agent is determined by observation of which side of the road already has more agents driving on it, within a particular area. In this respect, agents imitate what the majority of their neighbours are doing.

Similarly, Savarimuthu et al. [15] also use imitation in their work, which considers the *ultimatum game* in the context of providing advice to agents on whether to change their norms in order to enhance performance. In the ultimatum game, two agents must decide how to share a certain amount of money between them, starting with one agent offering a certain division of the money to the other. If the second agent agrees, then the money is divided between the agents according to the proposal, otherwise both agents gain nothing. Here, each agent has a personal norm that defines its proposal strategy. In addition, agents are able to request advice regarding their proposal strategy from only one agent, the *leader*, which is believed to have the best performance in the requesting agent's neighbourhood. Moreover, agents are capable of accepting of refusing the advice according to their autonomy level.

In relation to the learning aspects of our work, different forms of learning have also been used by other researchers. For example, Walker et al. [20] used a simple strategic update function in their model, based on Conte et al.'s [5] work. In their model, agents wander around searching for food in order to gain energy. However, since this movement causes them to lose energy, they need to find as much food as they can, and incurring the least movement in doing so. For this reason, agents follow different strategies, and change from one strategy to another according to a *majority rule*, which instructs an agent to switch to another strategy if it finds that the other strategy is used by more agents that its current strategy.

A more complex form of learning has been used by Mukherjee et al. [11, 17], who adopt Q-learning and some of its variants (WOLF-PHC and *fictitious play*) to show the effect of learning on norm emergence. They experimented with two different scenarios, first of homogeneous learning agents (where all agents have the same learning algorithm), and second heterogeneous learning agents (where agents can have different learning algorithms). Their results suggest that norm emergence is achieved in both situations, but is slower in heterogeneous environments.

In addition, some researchers (for example, [14, 19, 12]) have also considered the effect of various types of interaction networks on the *achievement* and *speed* of norm emergence, with results indicating that different types of networks give different outcomes. Though this is an interesting and valuable area to consider, it is outside the scope of this particular paper, so we say no more about it here. Nevertheless, our approach, as reported in the previous sections, is consistent with the broad approach taken by these previous effrorts in terms of analysing the different factors that affect norm emergence. Indeed, the aim of our work is to investigate the effects of metanorms on norm emergence, particularly when metanorms are integrated in a model that reflects key characteristics of distributed systems.

6 Conclusion

In systems of self-interested autonomous agents we often need to establish cooperative norms to ensure the desired functionality. Axelrod's work on norm emergence [2] gives valuable insight into the mechanisms and conditions in which such norms may be established. However, there are two major limitations. First, as Mahmoud et al. [10] have shown previously, and explained in detail, norms collapse even in the metanorms game for extended runs. Second, the model suffers from limitations relating to assumptions of omniscience. In response to this latter point of concern, this paper has (i) investigated those aspects of Axelrod's investigation that are unreasonable in real-world domains, and (ii) proposed *BV learning* as an alternative mechanism for norm establishment that avoids these limitations.

More specifically, we replaced the evolutionary approach with a learning interpretation in which, rather than remove and replicate agents, we allow them to learn from others. Two techniques were considered: copying from a single agent and copying from a group. The former suffers the same problems of long term norm collapse associated with Axelrod's approach [10] but, by avoiding strategies that only perform well in restricted settings, the latter addresses the problems and brings about norm establishment. In addition, we addressed Axelrod's assumption of omniscience, in which agents considering metapunishment are not explicitly required to *see* the original defection. By doing so, however, the metapunishment activity in the population, for stabilising an established norm, decreases and leads to norm collapse.

Since learning strategies from *others* (either individuals or groups) is unable to establish norms for cooperation (and is, in addition, unrealistic since it

assumes that agent strategies are not private), we have developed an alternative, *BV learning*, in which agents learn from their *own* experiences. Through this approach we have shown that not only is it possible to avoid the unrealistic assumption of knowledge of others' strategies, but also that by enabling individuals to incrementally change their strategies we can avoid norm collapse, even with observation constraints on metapunishment.

In term of future work, our aim is to focus on applying the model to interaction networks in order to analyse how different network structures can impact on the achievement of norm emergence. In particular, our current model is limited in that the algorithm relies on agents comparing their own score to the average score of all other agents to determine if learning is warranted. This constrains our move towards turning Axelrod's model into something more suitable for real-world distributed systems and, in consequence, we aim to enable agents to estimate their learning needs based on their own, individual, experience by monitoring their past performance. Moreover, we also plan to investigate the possibility of integrating *dynamic* punishments, rather than the current static ones (that are fixed regardless of what has happened), by which agents can modify the punishments they impose on others according to available information about the severity of violation, or according to whether the violating agent is a repeat offender, and if so, how many times.

References

1. Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Operationalisation of norms for usage in electronic institutions. In: AAMAS 2006: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 223–225. ACM (2006)
2. Axelrod, R.: An evolutionary approach to norms. American Political Science Review 80(4), 1095–1111 (1986)
3. Boman, M.: Norms in artificial decision making. Artificial Intelligence and Law 7(1), 17–35 (1999)
4. Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 1021–1026 (2001)
5. Castelfranchi, C., Conte, R.: Simulative understanding of norm functionalities in social groups. In: Gilbert, N., Conte, R. (eds.) Artificial Societies: The Computer Simulation of Social Life, pp. 252–267. UCL Press (1995)
6. de Pinninck, A.P., Sierra, C., Schorlemmer, W.M.: Friends no more: norm enforcement in multiagent systems. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 640–642 (2007)
7. Epstein, J.M.: Learning to be thoughtless: Social norms and individual computation. Computational Economics 18(1), 9–24 (2001)
8. Flentge, F., Polani, D., Uthmann, T.: Modelling the emergence of possession norms using memes. Journal of Artificial Societies and Social Simulation 4(4) (2001)
9. Galan, J.M., Izquierdo, L.R.: Appearances can be deceiving: Lessons learned reimplementing Axelrod's evolutionary approach to norms. Journal of Artificial Societies and Social Simulation 8(3) (2005)

10. Mahmoud, S., Griffiths, N., Keppens, J., Luck, M.: An analysis of norm emergence in axelrod's model. In: NorMAS 2010: Proceedings of the Fifth International Workshop on Normative Multi-Agent Systems. AISB (2010)
11. Mukherjee, P., Sen, S., Airiau, S.: Emergence of norms with biased interactions in heterogeneous agent societies. In: Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp. 512–515 (2007)
12. Nakamaru, M., Levin, S.A.: Spread of two linked social norms on complex interaction networks. Journal of Theoretical Biology 230(1), 57–64 (2004)
13. Riolo, R., Cohen, M., Axelrod, R.: Evolution of cooperation without reciprocity. Nature 414, 441–443 (2001)
14. Savarimuthu, B.T.R., Cranefield, S., Purvis, M., Purvis, M.: Norm emergence in agent societies formed by dynamically changing networks. In: IAT 2007: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 464–470 (2007)
15. Savarimuthu, B.T.R., Cranefield, S., Purvis, M., Purvis, M.: Role Model Based Mechanism for Norm Emergence in Artificial Agent Societies. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007 Workshops. LNCS (LNAI), vol. 4870, pp. 203–217. Springer, Heidelberg (2008)
16. Savarimuthu, B.T.R., Purvis, M., Purvis, M., Cranefield, S.: Social Norm Emergence in Virtual Agent Societies. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 18–28. Springer, Heidelberg (2009)
17. Sen, S., Airiau, S.: Emergence of norms through social learning. In: IJCAI 2007: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 1507–1512. Morgan Kaufmann Publishers Inc. (2007)
18. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
19. Villatoro, D., Sen, S., Sabater-Mir, J.: Topology and memory effect on convention emergence. In: Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technologies, pp. 233–240. IEEE (2009)
20. Walker, A., Wooldridge, M.: Understanding the Emergence of Conventions in Multi-Agent Systems. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi-Agent Systems, pp. 384–389. MIT Press (1995)
21. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning 8(3-4), 279–292 (1992)
22. López y López, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: AAMAS 2002: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 674–681. ACM (2002)
23. Yamashita, T., Izumi, K., Kurumatani, K.: An Investigation into the Use of Group Dynamics for Solving Social Dilemmas. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 185–194. Springer, Heidelberg (2005)

Establishing Norms for Network Topologies

Samhar Mahmoud[1], Nathan Griffiths[2], Jeroen Keppens[1], and Michael Luck[1]

[1] Department of Informatics
King's College London
London WC2R 2LS, UK
samhar.mahmoud@kcl.ac.uk
[2] Department of Computer Science
University of Warwick
Coventry CV4 7AL, UK

Abstract. In order to establish a norm in a society of agents, metanorms have previously been proposed as a means of ensuring not that norms are complied with, but that they are enforced. Yet while experimental results have shown that metanorms are effective in fully-connected environments such as that used by Axelrod, there has been limited consideration of such metanorm models with different but more realistic topological configurations. In this paper, therefore, we consider the use of metanorms in supporting norm establishment in lattices and small world networks. Our results suggest that norm establishment is achievable in lattices and small worlds.

1 Introduction

In peer-to-peer systems, agents share resources (hardware, software or information) with others, but if there is no cost to access files nor any limit on the number of files accessible, then there is no incentive to respond to requests nor, more generally, to establish cooperation in the system. Yet cooperation is needed: when self-interested autonomous agents must exchange information without any central control, non-compliance (due to selfish interests) can compromise the entire system. The use of *norms* to provide a means of ensuring cooperative behaviour has been proposed by many [3, 5, 6, 10, 13–15, 17] but, as shown by Axelrod [1], norms alone may not lead to the desired outcomes. In consequence, *metanorms* have been proposed as a means of ensuring not that norms are complied with, but that they are enforced. While experiments have shown that metanorms are effective in fully-connected environments as used by Axelrod, there has been limited consideration of metanorms with different but more realistic topological configurations, which fundamentally change the mechanisms required to establish cooperation.

Some work has already been undertaken on examining the impact of different topologies on norm establishment. For example, Savarimuthu et al. [9] consider the *ultimatum game* in the context of providing advice to agents on whether to change their norms in order to enhance performance for random and scale-free networks. Delgado et al. [4] study norm emergence in coordination games

S. Cranefield et al. (Eds.): COIN 2011, LNCS 7254, pp. 203–220, 2012.

in scale-free networks, and Sen et al. [11] examine rings and scale-free networks in a related context. Additionally, Villatoro et al. [14] explore norm emergence with memory-based agents in lattices and scale-free networks.

While these efforts provide valuable and useful results, the context of application has been limited, with only two agents involved in each encounter, rather than a larger population of agents. This simplifies the problem when compared with those in which the actions of multiple interacting agents can impact on norm establishment. In particular, Axelrod's seminal model [1] has provided the foundation for several investigations into norm emergence, yet offers a very general framework, comprising the use of norms and metanorms in populations of agents where the overall behaviour determines whether a norm is established. In this paper we extend Axelrod's model to address the context of different topological configurations.

The paper begins with an outline of Axelrod's metanorms game, adjusted to suit the purposes of this paper, and augmented with a learning mechanism. Section 3 then considers the problems that arise from the use of different topologies, and Sections 4 and 5 describe in detail the impact of applying the model in lattices and small worlds.

2 The Metanorms Game

Our model aims to simulate a realistic distributed system in which a community of self-interested agents is encouraged, without being instructed to do so by a central authority, to adhere to a behavioural constraint, or *norm*, that benefits the community but not the individual agent adhering to the norm. This simulation provides an experimental setting that enables us to test under what conditions a situation arises in which the norm governs the behaviour of individual agents.

2.1 Axelrod's Model

Inspired by Axelrod's model [1], our simulation focusses only on the essential features of the problem. In the simulation, the agents play a game iteratively; in each iteration, they make a number of binary decisions. First, each agent decides whether to comply with the norm or to defect. Defection brings a reward for the defecting agent, and a penalty to all other agents, but each defector risks being observed by the other agents and punished as a result. These other agents thus decide whether to punish agents that were observed defecting, with a low penalty for the punisher and a high penalty for the punished agent. Agents that do not punish those observed defecting risk being observed themselves, and potentially incur metapunishment. Thus, finally, each agent decides whether to metapunish agents observed to spare defecting agents. Again, metapunishment comes at a high penalty for the punished agent and a low penalty for the punisher.

The behaviour of agents in each round of the game is random, but governed by three variables: the probability of being seen S, boldness B, and vengefulness V. Each round agents are given a fixed number of opportunities o to defect or

Algorithm 1. The Simulation Control Loop: *simulation*$(T, H, P, E, \gamma, \delta)$

1. **for** each round **do**
2. interact(T, H, P, E)
3. learn(γ, δ)

comply, each of which has a randomly selected probability of a defection being seen. Boldness determines the probability that an agent defects, such that if an agent's boldness exceeds the probability of a defection being seen then the agent defects. Vengefulness is the probability that an agent punishes or metapunishes another agent. Thus the boldness and vengefulness of an agent are said to comprise that agent's strategy. After several rounds of the game, each agent's rewards and penalties are tallied, and successful and unsuccessful strategies are identified. By comparing themselves to other agents on this basis, the strategies of poorly performing agents are revised such that features of successful strategies are more likely to be retained than those of unsuccessful ones. We need not be concerned with the details of the learning algorithm in this paper, beyond the fact that boldness and vengefulness are simply revised upward or downward as appropriate, in line with a specified learning rate. If most agents employ a strategy of low boldness and high vengefulness, it can be argued that the norm has become *established* in that community, because strategies that lead to defection or to sparing defecting agents are unlikely and lead to high penalties.

2.2 Our Simulation Algorithm

Given Axelrod's model as a starting point, we have previously developed refinements of it that are better suited to real-world distributed systems, by not requiring agents to have information on the private strategies of others, and by allowing agents to improve performance, via a reinforcement learning technique. Since this is not the focus of this paper, we will not provide a full explanation; the full details of why and how are provided in a sister paper [8]. Nevertheless, since these refinements are the starting point for our work here, in this section we briefly review the presentation in [8] to set up subsequent sections.

First, in order to determine the unique effect of each individual action on agent performance, each agent keeps track of four different utility values: the *defection score* (DS) incurred by an agent who defects, the *punishment score* (PS) incurred by an agent who punishes or metapunishes another (as a result of an enforcement cost, and the *no punishment score* (NPS) incurred by an agent who does not punish another when it should, and is consequently metapunished. In addition these are combined into a total score (TS).

In this context, we can consider the algorithms used in our simulation, in two phases, as represented in Algorithms 2 and 3, called by Algorithm 1. More precisely, in Algorithm 2, each agent has various defection opportunities (o), and defects if its boldness is greater than the probability of its defection being seen. if an agent defects (Line 3), its DS increases by a *temptation payoff*, T

Algorithm 2. Interact(T, H, P, E)

```
1.  for each agent i do
2.    for each opportunity to defect o do
3.      if B_i > S_o then
4.        DS_i = DS_i + T
5.        for each agent j : j ≠ i do
6.          TS_j = TS_j + H
7.          if see(j,i,S_o) then
8.            if punish (j, i, V_j) then
9.              DS_i = DS_i + P
10.             PS_j = PS_j + E
11.           else
12.             for each agent k : k ≠ i ∧ k ≠ j do
13.               if see(k,j,S_o) then
14.                 if punish (k, j, V_j) then
15.                   PS_k = PS_k + E
16.                   NPS_j = NPS_j + P
```

(Line 4), but it *hurts* all others in the population, whose scores decrease by H (line 6), where H is a negative number that is thus added to the score. If an agent cooperates, no scores change. DS thus determines whether an agent should increase or decrease boldness in relation to its utility.

However, each hurt agent can in turn observe the defection and react to it with punishment that is probabilistic to its vengefulness. Punishment and metapunishment both have two-sided consequences: if an agent j sees agent i defect in one of its opportunities (o) to do so, with probability S_o (Line 7), and decides to punish it (which it does with probability V_j; Line 8), i incurs a punishment cost, P, to its DS (Line 9), while the punishing agent incurs an enforcement cost, E, to its PS (Line 10). Note that both P and E are negative values, so they are added to the total when determining an overall value. If j does not punish i, and another agent k sees this in the same way as previously (Line 13), and decides to metapunish (Line 14), then k incurs an enforcement cost, E, to its PS, and j incurs a punishment cost P to its NPS.

In the learning phase, in Algorithm 3, and as mentioned above, each agent uses the various scores to determine how to improve its actions in the future. At the beginning of the learning procedure, the agent calculates its total score by combining all the other scores. In order to ensure a degree of exploration (similar to mutation in the original model's evolutionary approach, to provide comparability), we adopt an *exploration rate*, γ, which regulates adoption of random strategies from the available strategies universe (Line 8).

If the agent does not explore, then if defection is the cause of a low score (Line 12), an agent decreases its boldness, and increases it otherwise. Similarly, agents increase their vengefulness if they find that the effect of not punishing is worse than the effect of punishing (Line 22), and decrease vengefulness if the situation is reversed. As both PS and NPS represent the result of two mutually

Algorithm 3. Learn(γ, δ)

```
1.  Temp = 0
2.  for each agent i do
3.     TS_i = TS_i + DS_i + PS_i + NPS_i
4.     Temp = Temp + TS_i
5.  AvgS = Temp/no_agents
6.  for each agent i do
7.     if TS_i < AvgS then
8.        if explore(γ) then
9.           B_i = random()
10.          V_i = random()
11.       else
12.          if DS_i < 0 then
13.             if B_i − δ < 0 then
14.                B_i = 0
15.             else
16.                B_i = B_i − δ
17.          else
18.             if B_i + δ > 1 then
19.                B_i = 1
20.             else
21.                B_i = B_i + δ
22.          if PS_i < NPS_i then
23.             if V_i − δ < 0 then
24.                V_i = 0
25.             else
26.                V_i = V_i − δ
27.          else
28.             if V_i + δ > 1 then
29.                V_i = 1
30.             else
31.                V_i = V_i + δ
```

exclusive actions, their difference for a particular agent determines the change to be applied to vengefulness. For example, if $PS > NPS$, then punishment has some value, and vengefulness should be increased. As indicated previously, this is covered in more detail in [8], but we will provide no further details here.

3 Imposing Topologies on Metanorms

Axelrod's model is interesting and valuable in examining how norms can be established in a population of agents. Using our simulation model, we are able to match Axelrod's results (and in fact improve on them, since Axelrod's model fails for extended runs of the simulation, as demonstrated by [7]). In a fully connected network (in which each agent is connected to every other agent),

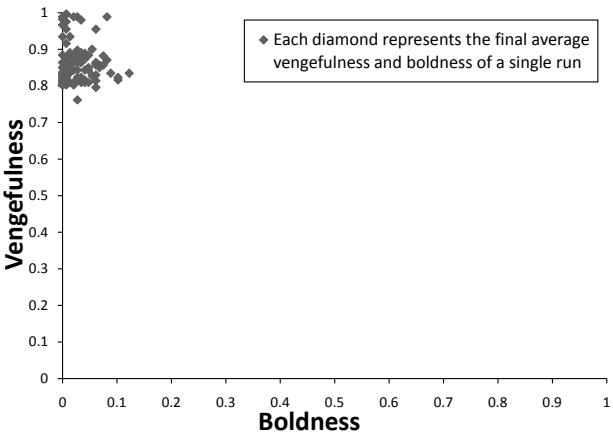
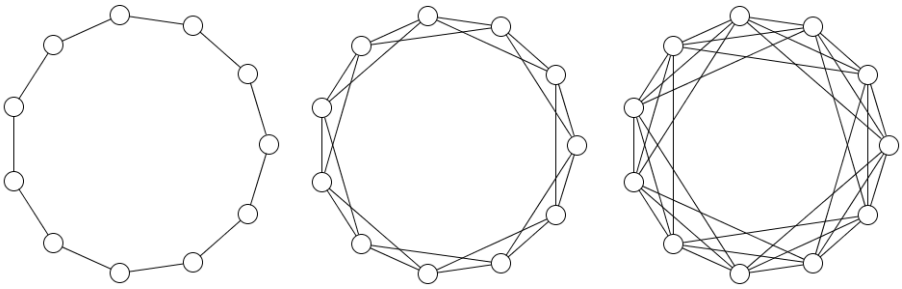


Fig. 1. Strategy improvement

matching Axelrod's initial configuration, we get the results shown in Figure 1, and detailed in [8]. This provides a valuable illustration of the value of norms and the use of metanorms to avoid norm collapse in a system in which there is no central control, but Axelrod's model omits consideration of some important aspects. In particular, in real-world computational domains, such as peer-to-peer and wireless sensor networks, the network of agents is not fully connected, with agents tending to interact with a small subset of others on a regular basis, yet it is only through such interactions that defection can be observed and punishment administered. Note that we restrict ourselves here to computational abstractions that apply to such environments rather than to physical or human networks.

Thus, while Axelrod's model assumes a fully connected network, an unlikely and unreasonable assumption, other network topologies must instead be considered, reflecting different potential configurations of agents, in which agents are connected only to a subset of other agents, their *neighbours*. This constraint on connectivity between agents implies some adjustments to Axelrod's model, as follows.

First, in Axelrod's model it is assumed that an agent's defection penalises all other agents in the population. The introduction of a topology enables us to restrict the penalty to only those agents with which the defector interacts. Second, in Axelrod's model, agents are assumed to be able to observe the entire population. By introducing a topology, we employ a more realistic model in which an agent can only observe those agents with which it interacts. Third, punishment requires observation of misbehaviour. In Axelrod's model, this requirement is implicit as it makes no meaningful distinction. However, by introducing constraints on observation and rendering the model more realistic, a further refinement is required: an agent can only punish a defector if the agent can observe the defector. In addition, an agent can only metapunish an agent that fails to punish



(a) NB Size 1 (b) NB Size 2 (c) NB Size 3

Fig. 2. Examples of lattice topologies

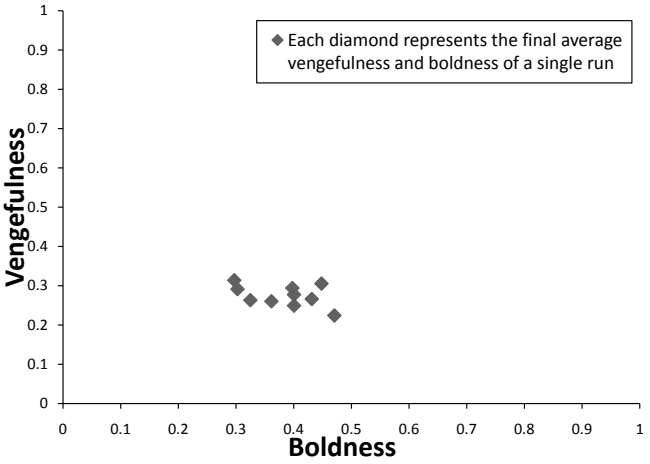
a defector if it can observe both the defector *and* the agent that fails to punish the defector. Finally, in order to enhance an agent's individual performance, it compares itself to others in the population before deciding whether to modify strategy. However, since agents can only observe their neighbours, these are the only agents they are able to learn from.

In consequence, the algorithms presented above are no longer adequate, and need to be changed as follows. First, in Algorithm 2, Line 5 needs to consider only agent i's neighbours rather than all of the agents in the population, and Line 12 needs to consider only agent j's neighbours. Then in Algorithm 3, the average score in Line 3, $AvgS$ should instead refer to the average score of the neighbourhood (that is, those agents to which agent i is connected. In this way, and with these simple modifications, our algorithms now address the needs of different topological structures.

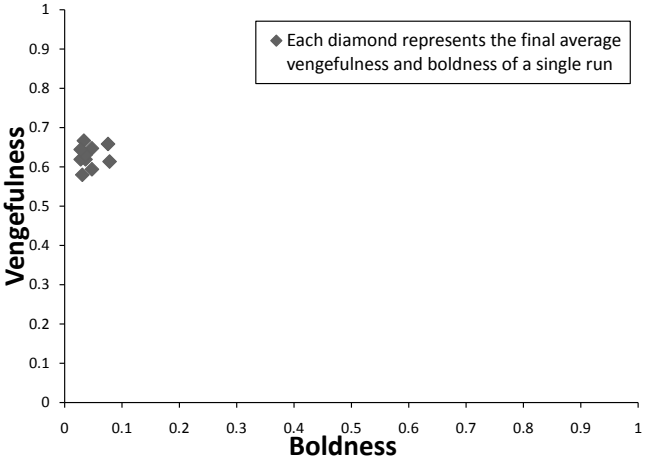
In what follows, we consider these modifications to the basic model in the context of different kinds of topologies, in particular small world models and scale-free networks. However, to start, we introduce lattices, since they provide the foundation on which small-worlds are based.

4 Metanorms in Lattices

A lattice (typically a simple ring structure) is perhaps the simplest network topology we consider, in particular, because it is also used as a base for more interesting and valuable topologies. In a (one-dimensional) lattice with neighbourhood size n, agents are situated on a ring, with each agent connected to its neighbours n or fewer hops (lattice spacings) away, so that each agent is connect to exactly $2n$ other agents. Thus, in a lattice topology with $n = 1$, each agent has two neighbours and the network forms a ring as shown in Figure 2(a). In a lattice topology with $n = 3$, each agent is connected to 6 neighbours, as shown in Figure 2(c).



(a) Lattice with neighbourhood size 1



(b) Lattice with neighbourhood size 3

Fig. 3. Smaller neighbourhoods in lattices

4.1 Neighbourhood Size

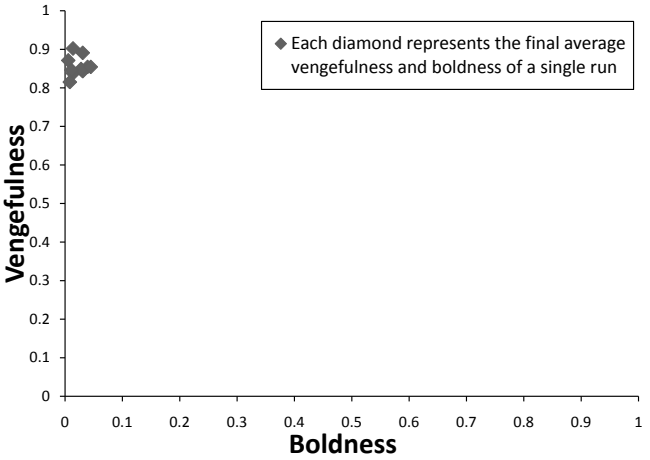
It is clear that, depending on the neighbourhood size, lattices may be more or less connected. Those with larger neighbourhood sizes are more similar to Axelrod's fully connected model; our hypothesis is that as the neighbourhood size increases, the greater connections between agents enable punishment and metapunishment to become more effective in reducing boldness and increasing vengefulness. In order to investigate this hypothesis, we ran several experiments.

In our first set of experiments, we used 51 agents (so we have an even number, plus one, to account for the $2n$ neighbours plus our original agent), and varied the neighbourhood size between the least connected lattice (the ring topology) and the most connected lattice ($n = 25$). Each experiment involved 10 separate runs, with each run comprising 1,000 timesteps. for a particular neighbourhood size.

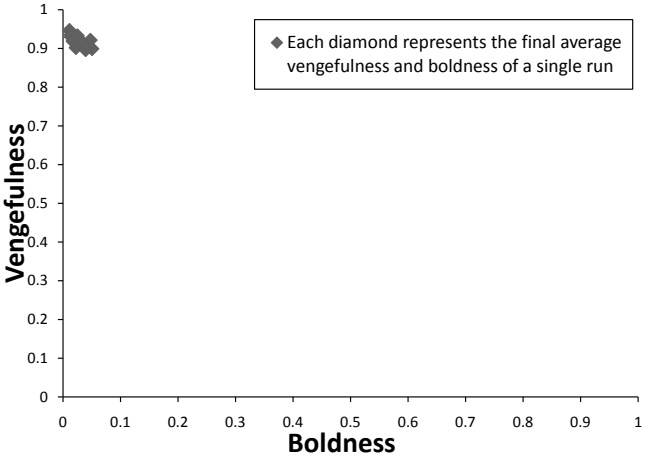
For the least connected lattice (n of 1), no norm is established, as runs ended in both relatively low boldness and relatively low vengefulness (see Figure 3(a)). In this case, though agents rarely defect, they also rarely punish a defection. This constitutes an unstable situation in which defecting could be a rewarding behaviour for agents as it is relatively unlikely to be penalised. However, increasing the neighbourhood size slightly to 3 (Figure 3(b)) has a noticeable impact on the results, as the boldness of the population drops almost to 0, which means that agents do not defect. While the level of vengefulness increases, it is still not at a level that can be considered to correspond to norm emergence, since agents might still not punish a defection without being metapunished for not doing so.

In addition, increasing the neighbourhood size to 13 has the same effect on boldness and a stronger effect on vengefulness (see Figure 4(a)), as vengefulness increases further, and almost to its maximum, of 1, when the neighbourhood size of 19 is used (see Figure 4(b)). These results suggest that increasing neighbourhood size strengthens norm emergence, by virtue of agents being more willing to punish norm violators. In seeking to provide more detail for analysis, the results of all runs were averaged, and shown on the graph in Figure 5, with neighbourhood size plotted against boldness and vengefulness. This shows that a neighbourhood size as small as 2 is enough to maintain boldness near 0, indicating that agents do not defect except when they *explore* as a result of sometimes adopting random strategies (introduced for comparability with Axelrod's model). Conversely, increasing the neighbourhood size has a major impact on vengefulness, until the neighbourhood size reaches around 15 (at which point an agent is connected to half the population) when it brings only very minor change. This is because, in a poorly connected environment, agents that do not punish defection can more easily escape metapunishment than in a more connected environment.

As we hypothesised, increasing neighbourhood size brings a corresponding effect on the strategy of agents (in terms of boldness and vengefulness). Only the most poorly connected lattices have moderate levels of boldness, with vengefulness increasing monotonically over a longer period before it stabilises at a level consistent with norm establishment. The connections between agents give rise



(a) Lattice with neighbourhood size 13



(b) Lattice with neighbourhood size 19

Fig. 4. Larger neighbourhoods in lattices

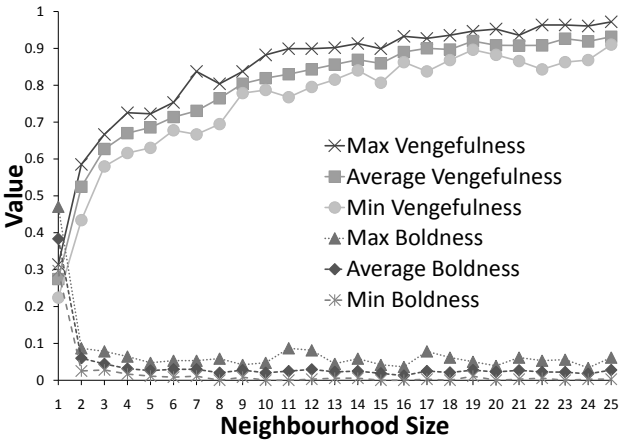


Fig. 5. Lattice: impact of neighbourhood size on final B and V

to this behaviour, with an increase in connections providing more opportunities for agents to respond to defectors appropriately.

4.2 Population Size

Now, if we increase the population size while keeping the neighbourhood size static, we decrease the relative number of connections among the overall population. This suggests that convergence to norm establishment should decrease, in line with the results obtained above. In the second set of experiments, therefore, the neighbourhood size was fixed and the population size varied between 51 and 1,001 agents. However, the results obtained, shown in Figure 6 for a neighbourhood size of 3 (though other values gave similar results), are not as expected, and suggest that increasing the population size has no effect on the rate of norm emergence, as all runs for all sizes of population end almost with the same level of boldness and vengefulness.

These results suggest that norm emergence in a community of agents that interact in a lattice is not affected by total population size but by neighbourhood size. By increasing the number of neighbours, norm establishment becomes more likely, irrespective of the size of the population. In other words, the likelihood of norm establishment is governed by the total amount of punishment that could potentially be brought upon a defector or an agent failing to punish a defector, which may be termed the *potential peer pressure* of a lattice. This is because such lattices essentially comprise multiple overlapping localities in which agents are highly connected: via punishments, the agents in these localities impose a strong influence on their neighbours. Increasing the population size simply increases the number of such overlapping regions.

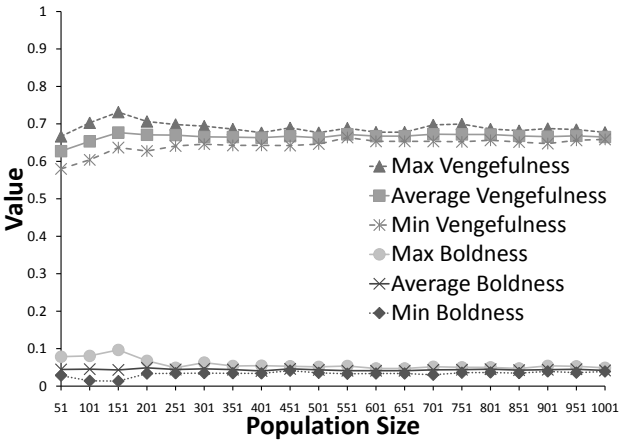


Fig. 6. Lattice: impact of population size on final B and V (neighbourhood size = 3)

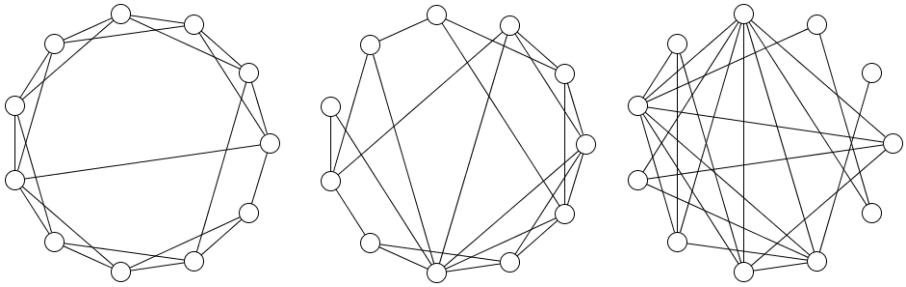
5 Metanorms in Small Worlds

While lattices are regular structures, as opposed to random structures, Watts and Strogatz noted that many biological, technological and social networks lie somewhere between the two: neither completely regular nor completely random [16]. They instead proposed *small world networks* as a variation of lattices in which agents are connected to others n or fewer hops (on the ring) away, but with some of the connections replaced by connections to other randomly selected nodes in the network, in line with some specific rewiring probability (RP).

Thus, while lattices essentially create overlapping localities of well connected agents (since agents are connected to $2n$ agents immediately surrounding them), the effect of small worlds is to break these connections. Though the number of connections does not change, the locality effect does, since there may no longer be localities of well connected agents, but instead agents with some connections to their local neighbours, and some connections to others elsewhere in the network. As these local regions break down, the strong influence of an agent's local neighbours, causing compliance with norms, should also break down because of the more sparse connections.

To verify this hypothesis, we investigated the impact of the rewiring probability by running the model with different values, in populations of 51 agents, for different neighbourhood sizes of 3 and 5. The results of the experiment with a neighbourhood size of 3 are shown in Figure 8, which indicates that increasing the RP decreases the final average vengefulness in the population. With a neighbourhood size of 5 the results are similar (not shown).

This is because, as a result of rewiring, agents no longer affect just their locality, but now affect agents that are much further away, consequently requiring



(a) RP = 0.1 (b) RP = 0.2 (c) RP = 0.8

Fig. 7. Examples of small worlds

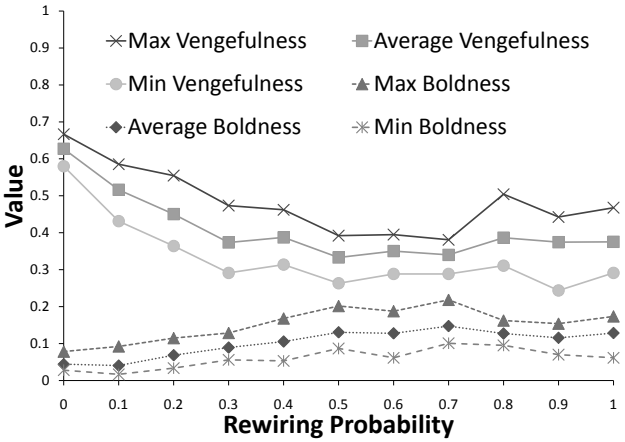


Fig. 8. Small world: impact of rewiring on final B and V (where neighbourhood size, $n = 3$)

establishment of the norm in multiple localities. For example, in the case of neighbourhood size of 3, it is clear that not only is the norm not established, but as the RP rises above small values, the trend moves further away from establishment, since the connections of agents are increasingly rewired, giving a locality effect similar to lattices with a neighbourhood size of 2 (discussed in Section 4.1). In addition, rewiring to other agents further away brings the need to establish the norm in all those localities to which an agent is connected, making it much more difficult.

In term of boldness, it is clear from the results that the RP of small worlds does not impact on the level of defection in the population since, independently, boldness remains very low, indicating that agents are very unlikely to defect.

5.1 Neighbourhood Size and Rewiring

As discussed in Section 4.1, increasing neighbourhood size causes an increase in vengefulness in lattices. In seeking to understand the impact in small worlds, we repeated the lattice experiments in this new context, for different values of the RP. Results for a rewiring probability of 0.4 are shown in Figure 9 (with results for other values of the RP being similar in trend), again showing that neighbourhood size increases vengefulness. However, note that, in comparison to lattices, vengefulness in small worlds is lower for the same neighbourhood size. This is because the agents must now respond to defections in different regions of the network, where there is less influence on behaviour, and thus potentially incurring greater enforcement costs.

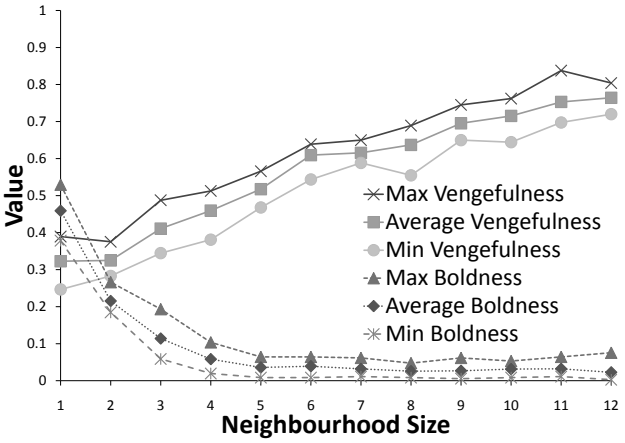


Fig. 9. Small world: impact of neighbourhood size on final B and V (RP=0.4)

5.2 Population Size and Rewiring Probabilities

Population has been shown to have no effect on norm establishment in lattices due to the *potential peer pressure* arising from the multiple overlapping localities. However, since these concentrated local regions of connected agents are weakened in small worlds, we repeated the previous experiments to determine the effect with RPs of 0.2, 0.4, 0.6, 0.8 and 1.0, and n of 5. The results indicate that boldness is not affected by the changes of the population size as it is always close to zero (not shown), but vengefulness decreases as the RP increases. More specifically, when the RP is 0.2, increasing the population size has little effect, as shown in Figure 10. However, for the other RP values, increasing the population size decreases vengefulness. Again, this is due to rewiring breaking down the strong locality effect, and this is magnified with increasing population sizes, since there is a greater opportunity for connections to other localities, causing a greater cost for agents seeking to bring about norm establishment in all these localities at once.

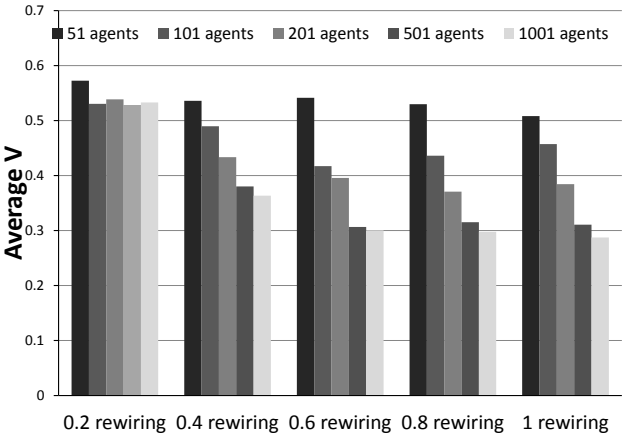


Fig. 10. Small worlds: impact of rewiring and population size on final vengefulness (where neighbourhood size $n = 5$)

6 Related Work

There has been much work that focusses on the issue of norm emergence in societies of interacting agents. However, most of this concentrates on analysing norm emergence over fully-connected networks [2, 12, 13, 15], and it was only relatively recently that attention shifted towards the effect of the structure of these societies. In this section, we review that part of the literature that does address these concerns.

In a particular effort, Delgado et al. [4] study the emergence of coordination in scale-free networks. Their study involves an interaction model of a multi-agent system, by which they try to analyse how fast coordination can spread among agents. Coordination here is represented through agents being in the same state, which is achieved when 90% of the agents do so. The framework they use is rather simple, however: an agent makes a choice between two different actions and they receive a positive payoff if they both choose the same action, or a negative payoff if their actions are different. Agents record the outcome of taking each of the two actions and pick the action with the better outcome for next interaction. The results of the work demonstrate that coordination can indeed be achieved over scale-free networks, but in a rather restricted setting.

Similarly, Sen et al. [11] use a game to investigate norm emergence over lattices and scale-free networks. In particular, they analyse the effect of increasing the number of actions available to agents, as well as the effect, on the speed of norm emergence, of increasing the number of agents in both scale-free networks and lattices. Their results suggest that both increasing the number of actions *and* increasing the number of agents causes a delay to the norm emergence in the population over a scale-free network. Similarly, norm emergence in lattices is

much slower when agents have a larger set of actions to choose from, or when the number of agents in the population is increased. Overall, their analysis shows that, for a small set of actions, it is faster for a norm to spread in a ring than in other topologies, followed by fully connected structures, and then scale-free networks. In contrast, for a large set of actions, it turns out that this is much faster in scale-free networks than in rings and fully connected structures.

As we have suggested, the models used in these previous pieces of work are relatively unsophisticated, with only two agents involved in an interaction, and reward values remaining fixed and not changing during the game. In response, Villatoro et al. [14] adopted the same concept of two-agent interactions, but introduced the notion of the reward of an action being determined through the use of the memory of agents, thus adding some dynamism to the model. Here, the reward of a certain action is determined by whether the action represents the majority action in both agents' memories, and the reward is proportional to the number of occurrences of this majority action in the their memories. However, it is not clear from where these rewards derive nor who applies them, as agents only have access to their memory. With regard to interaction networks, their work illustrates that increasing the neighbourhood size of a lattice accelerates norm emergence.In contrast, in the case of scale-free networks, norms do not emerge using the basic model. This is because of the development of *sub-conventions* that are persistent and hard to break, and which prevent the whole population from converging towards a single convention. A solution to this problem was found by giving *hub* agents (those with the majority of connections to others) more influence on the reward function.

Savarimuthu et al. [9] analyse the effect of advice on norm emergence over random and scale-free networks. For this reason, they use the *ultimatum game* in which two agents must decide how to share a certain amount of money. One agent offers a particular division of the money to the other and, if the second agent agrees, then the money is divided between the two agents according to this proposal. If the second agent does not agree, both agents gain nothing. Here, each agent has a personal norm that defines its proposal strategy and, in addition, agents are able to request advice about their proposal strategy from a *leader* agent that is believed to have the best performance in the neighbourhood. However, agents are capable of accepting or refusing the advice according to their autonomy level. The results obtained in this work show that norm emergence increases in speed over both random and scale-free networks with an increase in the average degree of connectivity.

Our work is rather different to these previous efforts in that we have investigated a more sophisticated model. In addition, we are not restricted to only two agents, and consider arbitrary numbers of them, since any agent's actions can be observed by all of its neighbours. These neighbours can in turn react by choosing to punish or to avoid doing so, potentially generating further metapunishments by other observing agents. Finally, sanctions applied in our case are dependent on the decisions of all agents that observe a violation, thus making them change with the number of agents involved.

7 Conclusions

In this paper, we have investigated mechanisms that encourage norms to emerge in communities of self-interested agents, without interference of a central or outside authority, under the realistic constraint that agents can only influence one another if they regularly interact. Based on Axelrod's seminal work, our model's substantial novel extension examines the impact of different types of topologies of interaction on norm emergence. Our results show that in circumstances in which *each* agent regularly interacts with a small number of other agents, as in lattices and small worlds, Axelrod's mechanisms to encourage norm emergence remain largely effective. More precisely, it is very effective for lattices, but its effectiveness varies with the rewiring probability in small worlds. Moreover, we have demonstrated that, given fixed penalties, for lattices, the effectiveness of Axelrod's approach only depends on the number of neighbours of each agent, *not* on the total population size. For small worlds, increasing the population size with a high rewiring probability decreases vengefulness, constraining norm emergence significantly. Thus, topology must be considered: in the case of a lattice or a small world, Axelrod's proposed approach will be effective for sufficiently large neighbourhood sizes.

References

1. Axelrod, R.: An evolutionary approach to norms. American Political Science Review 80(4), 1095–1111 (1986)
2. Boman, M.: Norms in artificial decision making. Artificial Intelligence and Law 7(1), 17–35 (1999)
3. de Pinninck, A.P., Sierra, C., Schorlemmer, W.M.: Friends no more: norm enforcement in multiagent systems. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 640–642 (2007)
4. Delgado, J., Pujol, J.M., Sangüesa, R.: Emergence of coordination in scale-free networks. Web Intelligence and Agent Systems 1, 131–138 (2003)
5. Epstein, J.M.: Learning to be thoughtless: Social norms and individual computation. Computational Economics 18(1), 9–24 (2001)
6. Flentge, F., Polani, D., Uthmann, T.: Modelling the emergence of possession norms using memes. Journal of Artificial Societies and Social Simulation 4(4) (2001)
7. Mahmoud, S., Griffiths, N., Keppens, J., Luck, M.: An analysis of norm emergence in axelrod's model. In: NorMAS 2010: Proceedings of the Fifth International Workshop on Normative Multi-Agent Systems. AISB (2010)
8. Mahmoud, S., Griffiths, N., Keppens, J., Luck, M.: Overcoming omniscience in axelrod's model. In: Proceedings of the IAT Workshop on Coordination, Organizations, Institutions and Norms (2011)
9. Savarimuthu, B.T.R., Cranefield, S., Purvis, M., Purvis, M.: Role Model Based Mechanism for Norm Emergence in Artificial Agent Societies. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007 Workshops. LNCS (LNAI), vol. 4870, pp. 203–217. Springer, Heidelberg (2008)
10. Savarimuthu, B.T.R., Purvis, M., Purvis, M., Cranefield, S.: Social Norm Emergence in Virtual Agent Societies. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 18–28. Springer, Heidelberg (2009)

11. Sen, O., Sen, S.: Effects of Social Network Topology and Options on Norm Emergence. In: Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., da Silva, V.T., Matson, E., Polleres, A. (eds.) COIN 2009. LNCS, vol. 6069, pp. 211–222. Springer, Heidelberg (2010)
12. Sen, S., Airiau, S.: Emergence of norms through social learning. In: IJCAI 2007: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 1507–1512. Morgan Kaufmann Publishers Inc. (2007)
13. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
14. Villatoro, D., Sen, S., Sabater-Mir, J.: Topology and memory effect on convention emergence. In: Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technologies, pp. 233–240. IEEE (2009)
15. Walker, A., Wooldridge, M.: Understanding the Emergence of Conventions in Multi-Agent Systems. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi-Agent Systems, pp. 384–389. MIT Press (1995)
16. Watts, D.J., Svaratz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998)
17. Yamashita, T., Izumi, K., Kurumatani, K.: An Investigation into the Use of Group Dynamics for Solving Social Dilemmas. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 185–194. Springer, Heidelberg (2005)

Author Index