@

The AAMAS'07 Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN)

Honolulu, Hawaii, USA
May 14, 2007
An international workshop of the COIN series

Preface

Multi-agent systems are often understood as complex entities where a multitude of agents interact, usually with some intended individual or collective purpose. Such a view usually assumes some form of structure, or set of norms or conventions that articulate or restrain interactions in order to make them more effective in attaining those goals, more certain for participants, or more predictable. The engineering of effective coordination or regulatory mechanisms is a key problem for the design of open complex multi-agent systems.

In recent years, social and organizational aspects of agency have become a major issue in MAS research. Recent applications of MAS on Web Services, Grid Computing and Ubiquitous Computing enforce the need for using these aspects in order to ensure social order within these environments. Openness, heterogeneity, and scalability of MAS pose new demands on traditional MAS interaction models. Therefore, the view of coordination and control has to be expanded to consider not only an agent-centric perspective but societal and organization-centric views as well.

The overall problem of analyzing the social, legal, economic and technological dimensions of agent organizations, and the co-evolution of agent interactions, provide theoretically demanding and interdisciplinary research questions at different levels of abstraction. Consequently, this workshop provides a space for the convergence of concerns and developments from MAS researchers that have been involved with these issues from the complementary perspectives of coordination, organizations, institutions and norms.

The COIN@AAMAS07 event is part of a workshop series that started in 2005, and since then has been continued with two editions per year. In 2005, ANIREM and OOOP workshops were held at AAMAS'05, while in 2006 the two editions of COIN were held at AAMAS'06 and at ECAI'06. The second COIN workshop in 2007 will be held at the "Multi-Agent Logics, Languages, and Organisations Federated Workshops" (MALLOW'007) in Durham.

Out of 15 submissions to the COIN@AAMAS07 workshop, 9 papers were selected for inclusion in the workshop proceedings and presentation at the workshop, an acceptance rate of 60 %. Paper presentations are limited to 20 minutes, plus 10 minutes for questions. The workshop programme also includes an invited talk on "Agents Organizations and Web Services" by Mike Huhns (University of South Carolina) and Munindar Singh (North Carolina State University) and a round table at the end of the day.

We would like to thank the COIN@AAMAS07 PC members and additional reviewers for their hard work, the authors for having submitted their work, and the attendees for choosing COIN@AAMAS07 among the many other interesting workshops at AAMAS07. We sincerely hope that you enjoy the presentations during the day, as well as the fruitful discussions about these exciting research topics.

Jaime Sichman and Sascha Ossowski
COIN@AAMAS07 Organizers

Organization

Sascha Ossowski	University Rey Juan Carlos, Madrid, Spain
Jaime Simao Sichman	University of Sao Paulo, Brazil

Steering Committee

Guido Boella	Italy
Olivier Boissier	France
Virginia Dignum	The Netherlands
Victor Lesser	USA
Pablo Noriega	Spain
Andrea Omicini	Italy
Sascha Ossowski	Spain
Julian Padget	UK
Jaime Sichman	Brazil

Program Committee

Alessandro Provetti	Università degli Studi di Messina (Italy)
Andrea Omicini	University of Bologna (Italy)
Anja Oskamp	Free University Amsterdam (The Netherlands)
Carl Hewitt	MIT (USA)
Catherine Tessier	ONERA (France)
Christian Lemaître	Universidad Autónoma Metropolitana (Mexico)
Danny Weyns	Katholieke Universiteit Leuven (Belgium)
Eric Matson	Wright State University (USA)
Eugenio Oliveira	Universidade do Porto (Portugal)
Fabiola López y López	Benemérita Universidad Autónoma de Puebla (Mexico)
Frank Dignum	Utrecht University (the Netherlands)
Gabriela Lindemann	Humboldt University & Berlin (Germany)
Guido Boella	University of Torino (Italy)
Holger Billhardt	University Rey Juan Carlos & Madrid (Spain)
Javier Vázquez-Salceda	Universitat Politècnica de Catalunya (Spain)
Jomi Fred Hubner	UFRB Blumenau (Brazil)
Juan Antonio Rodríguez Aguilar	IIIA-CSIC (Spain)
Julian Padget	University of Bath (UK)
Leendert van der Torre	University of Luxembourg (Luxembourg)
Liz Sonenberg	University of Melbourne (Australia)
Mario Verdicchio	Politecnico di Milano (Italy)
Michael Luck	University of Southampton (UK)
Nicoletta Fornara	Università della Svizzera Italiana (Switzerland)
Olivier Boissier	ENS Mines Saint-Etienne (France)
Olivier Gutknecht	LPL (France)
Pablo Noriega	IIIA-CSIC (Spain)
PInar Yolum	Bogazici University (Turkey)
Stephen Cranefield	U. Otago (New Zealand)
Ulises Cortés	Universitat Politècnica de Catalunya (Spain)
Vicent Botti	Universitat Politècnica de València (Spain)
Victor Lesser	University of Massachusetts-Amherst (USA)
Virginia Dignum	University of Utrecht (The Netherlands)
Wamberto Vasconcelos	University of Aberdeen (UK)
Yves Demazeau	LEIBNIZ (France)

Additional reviewers

Arianna Tocchio
Dimitri Melaye
Grégory Bonnet
Guillaume Piolle

Table of Contents

Role Model Based Mechanism For Norm Emergence In Artificial Agent Societies

Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam Purvis and Martin Purvis

Department of Information Science, University of Otago, Dunedin, P O Box 56, Dunedin, New Zealand
(tonyr,scranefield,tehrany,mpurvis)@infoscience.otago.ac.nz

Abstract. In this paper we propose a mechanism for norm emergence based on role models. The mechanism uses the concept of normative advice whereby the role models provide advice to the follower agents. Our mechanism is built using two layers of networks, the social link layer and the leadership layer. The social link network represents how agents are connected to each other. The leadership network represents the network that is formed based on the role played by each agent on the social link network. The two kinds of roles are leaders and followers. We present our findings on how norms emerge on the leadership network when the topology of the social link network changes. The three kinds of social link networks that we have experimented with are fully connected networks, random networks and scale-free networks.

1 Introduction

Norms are a widely observed mechanism for enforcing discipline and prescribing uniform behaviour in human societies. Norms specify the way the members of a society should behave and help societies to improve co-operation and collaboration among their members [1]. Some examples of norms in modern societies include the exchange of gifts at Christmas, tipping in restaurants and dinner table etiquette.

Norms have been so much a part of different cultures, it is not surprising that it is an active area of research in a variety of fields including Sociology, Economics, Biology and Computer Science. However, norms have been of interest to multi-agent researchers only for a decade now. Norms are of interest to the MAS researchers as software agents tend to deviate from these norms due to their autonomy. So, the study of norms has become crucial to MAS researchers as they can build robust multi-agent systems that comply to norms and also systems that evolve and adapt norms dynamically.

Our objective in this paper is to propose a mechanism based on role models for norm emergence using the concept of oblique norm transmission in artificial agent societies. We will demonstrate that our mechanism results in norm emergence (100% norm convergence) by using it on top of three kinds of network topologies.

2 Background

Due to multi-disciplinary interest in norms, several definitions for norms exist. Habermas [2], a renowned sociologist, identified norm regulated actions as one of the four action patterns in human behaviour. A norm to him means *fulfilling a generalized expectation of behaviour*, which is a widely accepted definition for social norms.

Many social scientists have studied why norms are adhered to. Some of the reasons for norm adherence include a) fear of authority or power b) rational appeal of the norms c) emotions such as shame, guilt and embarrassment that arise because of non-adherence and d) willingness to follow the crowd.

2.1 Normative multi-agent systems

Research on norms in multi-agent systems is fairly recent [3–5]. Norms in multi-agent systems are treated as constraints on behaviour, goals to be achieved or as obligations [6]. There are two main research branches in normative multi-agent systems. The first branch focuses on normative system architectures [7], norm representations [8], norm adherence and the associated punitive or incentive measures [9, 10]. The second branch of research is related to emergence of norms.

2.2 Related work on emergence of norms

The second branch of research on norms focuses on two main issues. The first issue is on norm propagation within a particular society. According to Boyd and Richerson [11], there are three ways by which a social norm can be propagated from one member of the society to another. They are a) Vertical transmission (from parents to offspring) b) Oblique transmission (from a leader of a society to the followers) and c) Horizontal transmission (from peer to peer interactions).

Norm propagation is achieved by spreading and internalization of norms [4, 12]. Boman and Verhagen [4, 12] have used the concept of normative advice (advice from the leader of a society) as one of the mechanisms for spreading and internalizing norms in an agent society. The concept of normative advice in their context is based on an assumption that the norm has been accepted by the top level enforcer, the Normative Advisor, and the norm does not change. But, this context cannot be assumed for scenarios where norms are being formed (when the norms undergo changes).

So, the second issue that has received less attention is the emergence of norms. However, there is abundant literature in the area of sociology on why norms are accepted in agent societies and how they might be passed on. Karl-Dieter Opp [13] has proposed a theory of norm emergence from a sociology perspective. Epstein [14] has proposed a model of emergence based on the argument that the norms reduce individual computations.

The treatment of norms has been mostly in the context of an agent society where the agents interact with all the other agents in the society [4, 12]. Few researchers have considered the actual topologies of the social network for norm

emergence [15]. We consider that social networks are of importance to the emergence of norms as they provide the topology and the infrastructure on which the norms can be exchanged. We are inspired by previous works on the spreading of ideas (opinion dynamics [16]) over different network topologies.

Social networks are important for norm emergence because in the real world, people are not related to each other by chance. They are related to each other through the social groups that they are in, such as the work group, church group, ethnic group and the hobby group. Information tends to percolate among the members of the group through interactions. Also, people seek advice from a close group of friends and hence information gets transmitted between the members of the social network. Therefore, it is important to test our mechanism for norm emergence on top of social networks, a topic which is receiving attention among multi-agent researchers recently [15].

2.3 Social network topologies

In this section we describe three network topologies that we have considered for experimenting with norm emergence.

a) Fully connected network: In the fully connected network topology, each agent in the society is connected to all the agents in a given society. Many multi-agent researchers have done experiments with this topology. Most of their experiments involve interactions with all the agents in the society [4, 12].

b) Random network : Erdös and Renyi have studied the properties of random graphs and have demonstrated a mechanism for generating random networks [17]. An undirected graph G(n,p) has n vertices in which the edges are connected to each other with a probability p. It should be noted that the random network becomes fully connected network when p=1.

c) Scale-free network: Nodes in a scale-free network are not connected to each other randomly. Scale-free networks have a few well connected nodes called hubs and a large number of nodes connected only to a few nodes. This kind of network is called scale-free because the ratio of well connected nodes to the number of nodes in the rest of the network remains constant as the network changes in size. Figure 1 is an example of an Albert-Barabasi scale-free network where the size of the network is 50.

Albert and Barabasi [18] have demonstrated a mechanism for generating a scale-free topology based on their observations of large real-world networks such as the Internet, social networks and protein-protein interaction networks [19]. They have proposed a mechanism for generating scale-free networks based on the preferential attachment of nodes. At a given time step, the probability (p) of creating an edge between an existing vertex (v) and the newly added vertex is given by the following formula:

$$p = (\text{degree}(v)) \ / \ (|E| + |V|)$$

where (|E| and |V| respectively are the number of edges and vertices currently in the network (counting neither the new vertex nor the other edges that are being attached to it).

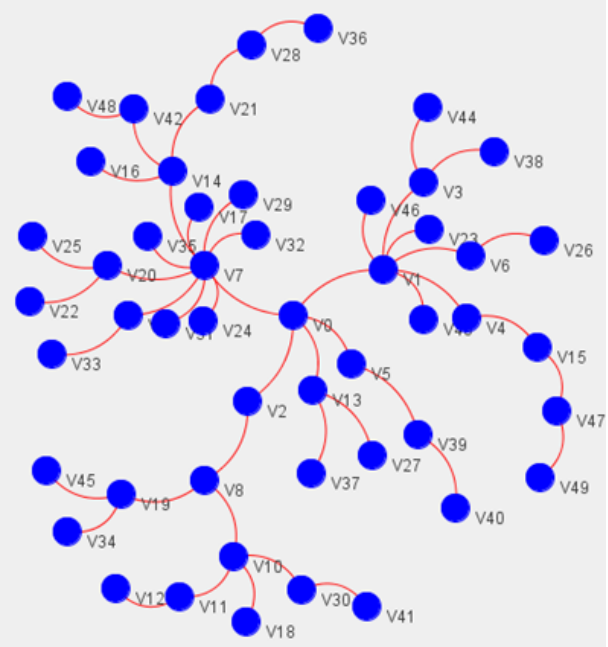


Fig. 1: An Albert-Barabasi scale-free network with 50 nodes

One may observe that the network shown in Figure 1 has a few well connected nodes, which are called hubs, e.g. vertices V7, and V1. A large number of nodes are connected to very few nodes. Scale-free networks exhibit a power law behaviour [18] where the probability of the existence of a node with k links (P(k)) is directly proportional to $k^{-\alpha}$ for some α.

Some characteristics of networks : Researchers have studied several characteristics of networks such as diameter (D), average path length (APL), degree distribution (k), clustering coefficient (C) etc. For our experiments we have used three of these characteristics whose definitions are given below.

- Degree distribution (k) : The degree of a node in an undirected graph is the number of incoming and outgoing links connected to particular node.
- Average Path Length (APL) : The average path length between two nodes is the average length of all possible paths between two nodes.
- Diameter (D) : The diameter of a graph is the longest path between any two nodes.

3 Role Model Agent Mechanism

In this section we describe a mechanism that facilitates norm emergence in an agent society. We have experimented with agents that play the Ultimatum game.

The context of interaction between the agents is the knowledge of the rules of the game. This game has been chosen because it is claimed to be sociologists' counter argument[1] to the economists' view on rationality [21]. In this context, when agents interact with each other, their individual norms might change. Their norms may tend to emerge in such a way that it might be beneficial to the societies involved.

3.1 The Ultimatum game

The Ultimatum game [22] is an experimental economics game in which two parties interact anonymously with each other. The game is played for a fixed sum of money (say x dollars). The first player proposes how to share the money with the second player. Say, the first player proposes y dollars to the second player. If the second player rejects this division, neither gets anything. If the second accepts, the first gets x-y dollars and the second gets y dollars.

3.2 Description of the multi-agent environment

An agent society is made up of a fixed number of agents. They are connected to each other using one of the social network topologies (fully connected, random or scale-free).

Norms in the agent society - Each agent in a society has an internal norm. Each agent also has a norm to represent its maximum and minimum proposal and acceptance values when playing the ultimatum game. This norm is called as the personal norm (P norm). A sample P norm for an agent is given below where min and max are the minimum and maximum values when the game is played for a sum of 100 dollars.

- Proposal norm (min=1, max=30)
- Acceptance norm (min=1, max=100)

The representations given above indicate that the proposal norm of an agent ranges from 1 to 30 and the acceptance norm of the agent ranges from 1 to 100.

The proposal norm initialized using a uniform distribution within a range of 1 to 100, is internal to the agent. It is not known to any other agent. The agents in a society are initialized with an acceptance norm that indicates that any agent which proposes within the range specified by the norm will be accepted.

[1] Sociologists consider that the norms are always used for the overall benefit of the society. Economists on the other hand state that the norms exist because they cater for the self-interest of every member of the society and each member is thought to be rational [20]. When Ultimatum game was played in different societies, researchers have observed that the norm of fairness evolved. As the players in this game choose fairness over self-interest, Sociologists' argue that, this game is the counter argument to economists' view on rationality.

The agents are only aware of their acceptance norms and are not aware of the acceptance norms of the other agents. In order to observe how proposal norms emerge, we assign a fixed value for acceptance norm to all the agents in the society. The acceptance norm of a society is given below.

– Acceptance norm (min=45, max = 55)

3.3 The norm emergence mechanism

The role models are agents who the societal members may wish to follow. The inspiration is derived from human society where one might want to use successful people as a guide. Any agent in the society can become a role model agent if some other agent asks for its advice. The role model agent represents a role model or an advisor who provides normative advise to those who ask for help. In our mechanism, each agent will have atmost one leader.

An agent will choose its role model depending upon the performance of its neighbours. We assume that agents that are connected know each other's performances. This is based on the assumption that people who are successful in the neighbourhood are easily recognizable. We argue that their success can be attributed to their norms.

Autonomy is an important concept associated with accepting or rejecting request to become a leader. When an agent is created, it has an autonomy value between 0 and 1. Depending upon the autonomy value, an agent can either accept or reject a request from another agent. Once rejected, an agent will contact the next best performing agent amongst its neighbours. Autonomy of an agent is also related to accepting or rejecting the advice provided by the leader agent.

Assume that agent A and B are acquaintances (are connected to each other in a network). If agent A's successful proposal average is 60% and agent B's successful proposal average is 80%, then agent A will send a request to agent B asking for its advice. If agent B accepts this request, B becomes the role model of agent A and sends its P norm to agent A. The agent is autonomous to choose or ignore the advice depending upon its autonomy. When agent A decides to follow the advice provided it modifies its P norm based on the advice received from its role model agent.

Figure 2 depicts the two layers of networks that are used in our mechanism. The circles represent agents. The solid lines represent the social link network also known as an acquaintance network.

In our mechanism, an agent plays a fixed number of Ultimatum games with each of its neighbours (agents that are linked to it). In total, highly connected agents play more games than the poorly connected agents. Highly connected agents benefit from playing more games because they retain their competitive advantage of obtaining a wide range of information or norms from the agents that they are connected to while the poorly connected agents rely on the information from one or two agents that they are connected to. A highly connected agent is more likely to know about the best norm earlier than the poorly connected agent.

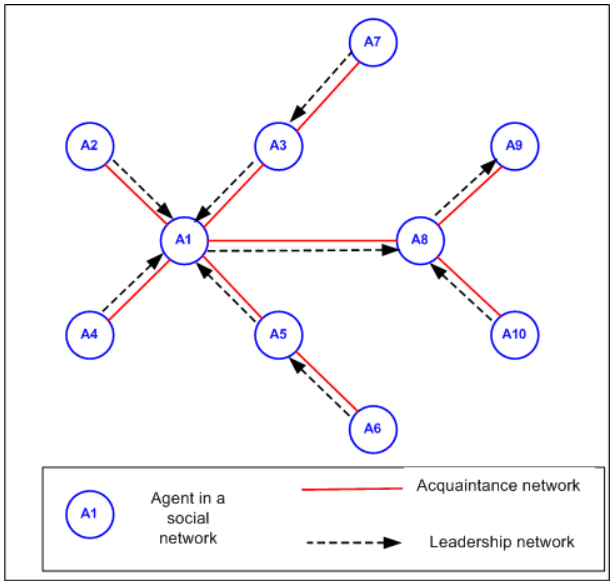


Fig. 2: Two layers of networks used in role model agent mechanism

After one iteration, every agent looks for the best performing player in its neighbourhood. After finding the best performing player, the agent sends a request to the player requesting the agent to be its role model or leader. If the requested agent decides to become the role model, it sends its P norm (normative advice) to the requester (follower agent). The follower agent modifies its norm by moving closer to the role model agent's norm. The dotted line with an arrow (directed line) represents the leadership network that emerges at the end of interactions. In Figure 2, A1 is the leader of A2,A3,A4 and A5. Arrows from these four agents point to A1. This new kind of network that emerges on top of the acquaintance network is called a leadership network.

4 Experiments and results

In this section we present the experiments that we undertook to demonstrate that our mechanism leads to complete norm emergence when tested on top of different kinds of network topologies.

4.1 Norm emergence on top of random and scale-free networks

The role model agent based mechanism norm propagation was evaluated using Erdös-Renyi (ER) random network and Albert-Barabasi (AB) scale-free network.

At first we studied the effects of changing the average degree of connectivity (<k>) on norm emergence, while maintaining a constant population size (N).

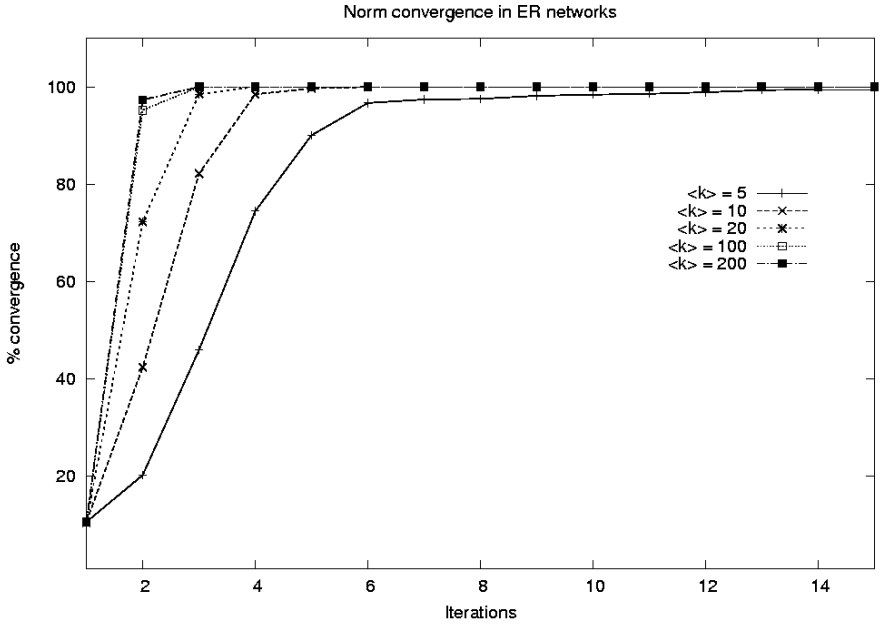


Fig. 3: Norm convergence in ER networks when average degree of connectivity is varied

We varied the degree of connectivity for the ER and AB networks with N=200 . It can be observed from Figure 3 that as <k> increased the rate of convergence increased in ER networks. When <k> is 10, 100% norm emergence was observed in the 6th iteration while it only took 3 iterations for convergence when the value of <k> is 200. Note that when <k> equals N, the network is fully connected, hence the convergence is faster. Similar results were also observed for AB networks (not shown here).

The comparison of ER and AB networks for the same values of N and <k> is shown in Figure 4. It can be observed that there is no significant difference in the rate of convergence in ER and AB networks. Our experimental results on norm convergence are in agreement with the statistical analysis carried out by Albert and Barabasi on the two kinds of networks [23]. They have observed that the diameter (D) and average path lengths (APL) of both the networks are similar for fixed values of N and <k>. The diameters of ER and AB networks, when N and <k> are fixed are directly proportional to log(N). As the diameters of both the networks are the same, the rate of norm convergence are similar.

The parameters D and APL of these networks decrease when the average connectivity of the network increases. When the average connectivity increases, it is easier for an agent to find a leader agent whose performance scores are high. If the average connectivity is low, it would take an agent a few iterations before its leader obtains the norm from a better performing agent. This explains why

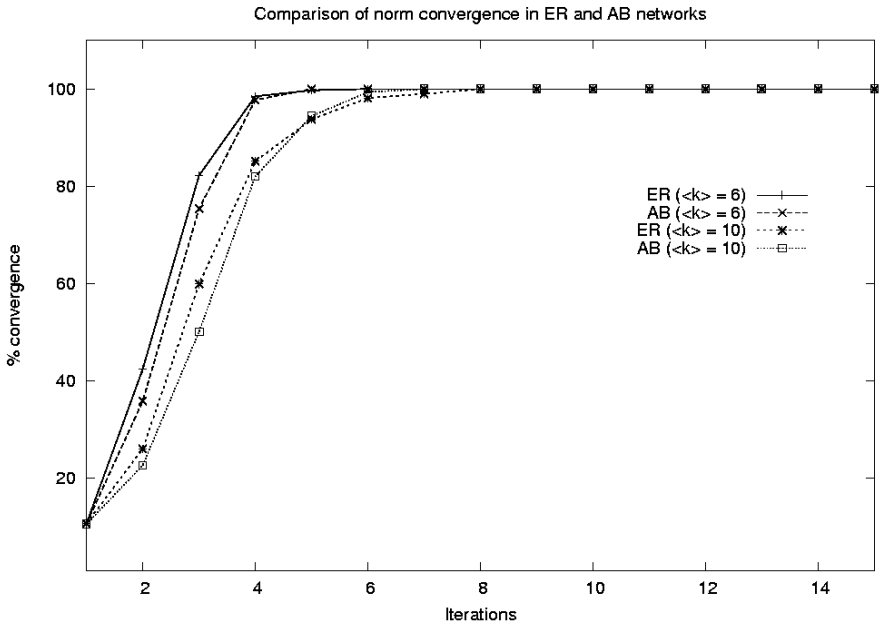


Fig. 4: Comparison of norm convergence in random vs scale-free networks

norm convergence is slower when average connectivity <k> decreases (shown in Figure 3).

Even though the norm emergence properties of both kinds of networks are comparable, it can be argued that the scale-free network is better suited to model norm propagation because in the real world, people are related to each other through the social groups that they are in, such as the work group and church group. Information percolates among the members of the group through interactions. Also, people seek advice from a close group of friends and hence information gets transmitted across social network. Other researchers have demonstrated that scale-free networks are well suited to explain mechanisms of disease propagation and dissemination of ideas [19]. Scale-free networks are more robust than random networks when random nodes start to fail and this phenomenon has been observed in real world networks [24].

Recently [25], it has also been observed that the diameter and average path lengths of an AB network depends upon the value of m. m is a constant that indicates the number of nodes to which a new node entering the network should be connected to, using the preferential attachment scheme. When m=1, D and APL are directly proportional to log(N) and for m>1, D is directly proportional to log(N)/log(log(N)). In this light, Albert and Barabasi have suggested that the scale-free networks should be more efficient in bringing nodes closer to each other which will be suitable for propagation of ideas and norms.

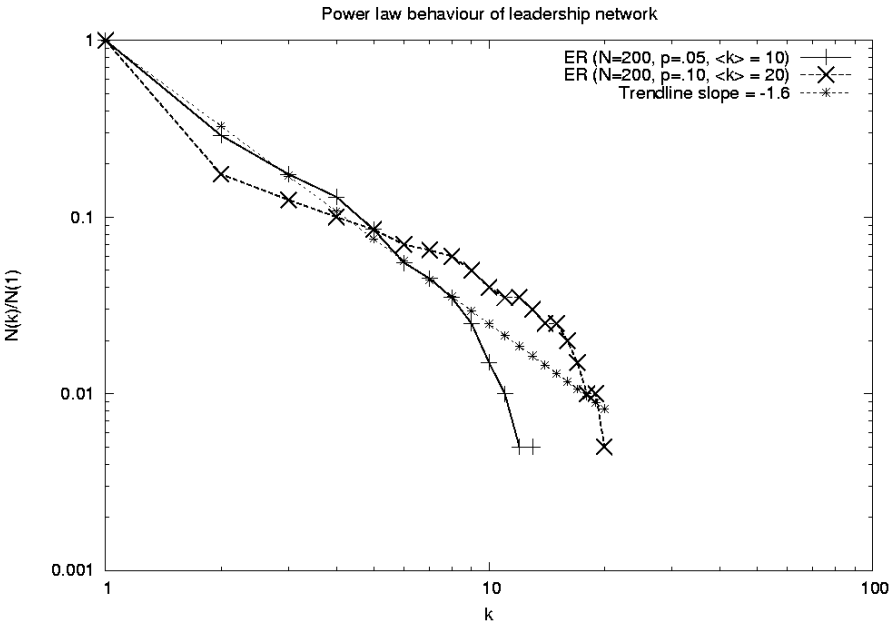


Fig. 5: Power law behaviour of the leadership network

4.2 Power law behaviour of the leadership network

We have also observed that the leadership network that emerges on top of the AB network follows power law behaviour. It is interesting to note that the leadership network that emerges on top of ER network follows power law behaviour when the average degree of connectivity is small. For smaller probabilities (p=.05, .1) we have observed that there are fewer leader agents with large number of followers and a large number of leaders with a few followers. Figure 5 shows the log-log plot of leaders with k followers in the x-axis and the number of leaders with k followers (N(k)) divided by the number of leaders with exactly one follower (N1) in the y-axis. The trendline shows the approximate power law behaviour of the leadership network. The slope of the power law curve was found to be -1.6. Our results are in agreement with that of Anghel et al. [26] who studied the emergence of scale-free leadership structures using minority game. In their work, an agent sends its game strategy to all the agents in its neighbourhood. There is no explicit notion of leadership as each agent maintains an internal model of who its leader is. In our work, each agent chooses its leader explicitly and the leader sends the norms only to its followers. Also, the agents in our model have the notion of autonomy which is more representative of a realistic society.

5 Discussion

Our work is different (see Section 2.3) from other researchers in this area as we use the concepts of oblique transmission in the mechanism we have proposed. Verhagen's thesis [12] focuses on the spreading and internalizing of norms. This assumes that a norm is agreed or chosen by a top level entity (say, a Normative Advisor) and this group norm (G norm) does not change. The G norm is spread to the agents through the normative advice using a top-down approach. Our work differs from this work as we employ a bottom-up approach. In our approach the P norm evolves continuously. In his work, the P norm changes to accommodate the predetermined group norm. Another important distinction is the consideration of network topologies in our work.

The experiments described in this paper are our initial efforts in the area of norm emergence. The experiments are limited to a single agent society. We are interested in experimenting with scenarios that involve two or more inter-linked societies. In the real world, we attach more weight to a particular person's advice than others. Similarly, the weights of the edges (links) should be considered when the agent makes a decision on who to choose as a role model agent. We plan to incorporate this idea in our future experiments. Also, addition or deletion of links to a given topology have not been considered in the current mechanism. This is analogous to people relocating and forming new links. We have planned to experiment with our mechanism on top of dynamically changing networks. We also intend to demonstrate that our mechanism is scalable.

6 Conclusions

We have explained our mechanism for norm emergence in artificial agent societies that is based on the concept of role models. We have demonstrated the use of oblique norm transmission for norm emergence. Our mechanism was tested on top of three network topologies. We have shown through our experimental results that complete norm emergence can be achieved using our proposed mechanism. We have compared our work with the researchers in this area and also discussed the future work.

References

1. Boella, G., Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computational and Mathematical Organization Theory **12**(2-3) (2006) 71–79
2. Habermas, J.: The Theory of Communicative Action : Reason and the Rationalization of Society. Volume 1. Beacon Press (1985)
3. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: Off-line design. Artificial Intelligence **73**(1-2) (1995) 231–252
4. Boman, M.: Norms in artificial decision making. Artificial Intelligence and Law **7**(1) (1999) 17–35
5. Conte, R., Falcone, R., Sartor, G.: Agents and norms: How to fill the gap? Artificial Intelligence and Law **7**(1) (1999) 1–15

6. Castelfranchi, C., Conte, R.: Cognitive and social action. UCL Press, London (1995)
7. López y López, F., Márquez, A.A.: An architecture for autonomous normative agents. In: Fifth Mexican International Conference in Computer Science (ENC'04), Los Alamitos, CA, USA, IEEE Computer Society (2004) 96–103
8. Boella, G., van der Torre, L.: An architecture of a normative system: counts-as conditionals, obligations and permissions. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems, AAMAS, New York, NY, USA, ACM Press (2006) 229–231
9. Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Operationalisation of norms for usage in electronic institutions. In: Proceedings of The Fifth International Joint Conference on autonomous Agents and Multi Agent Systems, AAMAS, New York, NY, USA, ACM Press (2006) 223–225
10. Axelrod, R.: An evolutionary approach to norms. The American Political Science Review **80**(4) (1986) 1095–1111
11. Boyd, R., Richerson, P.J.: Culture and the evolutionary process. University of Chicago Press, Chicago (1985)
12. Verhagen, H.: Norm Autonomous Agents. PhD thesis, Department of Computer Science, Stockholm University (2000)
13. Opp, K.D.: How do norms emerge? An outline of a theory. Mind and Society **2**(1) (2001) 101–128
14. Epstein, J.M.: Learning to be thoughtless: Social norms and individual computation. Computational Economics **18**(1) (2001) 9–24
15. Pujol, J.M.: Structure in Artificial Societies. PhD thesis, Llenguatges i Sistemes Informátics, Universitat Politénica de Catalunya (2006)
16. Fortunato, S.: Damage spreading and opinion dynamics on scale free networks (2004)
17. Erdös, P., Renyi, A.: On random graphs i. Publicationes Mathematicae Debrecen **6**(290) (1959)
18. Barabsi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286** (October 1999) 509–512
19. Mitchell, M.: Complex systems: Network thinking. Artificial Intelligence **170**(18) (2006) 1194–1212
20. Gintis, H.: Solving the Puzzle of Prosociality. Rationality and Society **15**(2) (2003) 155–187
21. Elster, J.: Social norms and economic theory. The Journal of Economic Perspectives **3**(4) (1989) 99–117
22. Slembeck, T.: Reputations and fairness in bargaining - experimental evidence from a repeated ultimatum game with fixed opponents. Technical report, EconWPA (1999) available at http://ideas.repec.org/p/wpa/wuwpex/9905002.html.
23. Albert, R., Barabasi, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics **74** (2002) 47–97
24. Albert, R., Jeong, H., Barabasi, A.L.: Error and attack tolerance of complex networks. Nature **406**(6794) (July 2000) 378–382
25. Bollobás, B., Riordan, O.: The diameter of a scale-free random graph. Combinatorica **24**(1) (January 2004) 5–34
26. Anghel, M., Toroczkai, Z., Bassler, K.E., Korniss, G.: Competition-driven network dynamics: Emergence of a scale-free leadership structure and collective efficiency. Physical Review Letters **92**(5) (2004) 0587011–0587014

Towards a Framework for Agent Coordination and Reorganization, AgentCoRe *

Mattijs Ghijsen, Wouter Jansweijer, and Bob Wielinga

Human Computer Studies Laboratory, Institute of Informatics, University of Amsterdam, email:{mattijs, jansw, wielinga}@science.uva.nl

Abstract. Research in the area of Multi-Agent System (MAS) organization has shown that the ability for a MAS to adapt its organizational structure can be beneficial when coping with dynamics and uncertainty in the MASs environment. Different types of reorganization exist, such as changing relations and interaction patterns between agents, changing agent roles and changing the coordination style in the MAS. In this paper we propose a framework for agent **Co**ordination and **Re**organization (AgentCoRe) that incorporates each of these aspects of reorganization. We describe both declarative and procedural knowledge an agent uses to decompose and assign tasks, and to reorganize. The RoboCupRescue simulation environment is used to demonstrate how AgentCoRe is used to build a MAS that is capable of reorganizing itself by changing relations, interaction patterns and agent roles.

1 Introduction

The quality of organizational design of a MAS has a large influence on its performance. However, this is not the only factor that determines MAS performance. It is the combination of the organizational design together with the nature of the task performed by the MAS and the characteristics of the environment in which the MAS is embedded that determines the performance of a MAS [1]. A MAS that operates in a dynamic environment can mitigate or reduce negative effects of dynamics in this environment by changing its organization [2].

In this paper we present the architecture of a framework that enables agents in a MAS to coordinate and reorganize. The goal of such an architecture is not to improve existing work on coordination by providing more efficient task/goal decomposition or task allocation, but rather to integrate several different aspects of reorganization into a single framework. To ensure a generic design, we describe our framework at the knowledge level [3], by providing the knowledge items and inferences an agent requires to coordinate and reorganize.

The organizational design of a MAS involves many aspects such as authority relations between agents, interaction patterns, agent roles and coordination

* The research reported in this paper is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant BSIK03024.

style. Research by Mintzberg has shown that the structure of human organizations and the coordination mechanisms used by its managers are closely related [4]. This, and other notions from the field of organizational design have already been applied in the area of MAS design [5, 6]. Since organization design and coordination are so closely related we believe that a framework for agent coordination should also provide the ability to reorganize.

Before we describe the AgentCoRe framework we discuss theory and related work on MAS reorganization. After a description of AgentCoRe, we will show how AgentCoRe is used to design and implement a MAS in the RoboCupRescue simulator [7]. We end with discussion, conclusions and directions for future work.

2 Theory and Related Work

In this paper we use definitions based on [8] and [5]. A task is defined as an activity performed by one or more agents to achieve a certain effect or goal in the environment. A task can be decomposed into subtasks and, in the case a task cannot be decomposed any further, it is called a primitive task. We define a role as a set of tasks that an agent is committed to perform when it is enacting that role. Capabilities are defined as a set of roles the agent is capable of enacting.

We define a MAS organization as a group of distributed agents, pursuing a common goal. The design of a MAS organization consists of relationships and interactions between the agents [9], agent roles [5] and coordination style [10]. Thus we define reorganization of a MAS as changing one or more of these organizational aspects. We assume that reorganization is triggered by the agents of the MAS, and not by a system designer or "human in the loop" as in [11].

A generic definition of coordination is given by [12] who define coordination as managing dependencies between activities. Research in the area of coordination in MAS has resulted in frameworks such as GPGP/TÆMS [13] and COM-MTDP [14] and both have been used in research on reorganization.

Nair et al. [15] extend [14] and change the composition of teams of agents to perform a rescue task in a highly dynamic environment where tasks can (de)escalate in size and new tasks are formed. Horling and Lesser change the relations and interaction patterns between agents in TÆMS structures but do not allow for role changes [16]. Their work is recently being extended by Kamboj and Decker [17] who use agent cloning [18] to allow for role changes in the organization. Barber and Martin present dynamic adaptation of coordination mechanisms as a mechanism for dealing with a dynamic environment [19].

The approaches described above all involve different aspects of reorganization; changing relations and interactions, changing agent roles and changing coordination style. In the next section we present the AgentCoRe framework which gives a knowledge level description of a framework for coordination and reorganization. We extend existing work by incorporating all aspects of reorganization described above into a single framework. By giving a knowledge level description, we refrain from computational details and focus on the required knowledge and reasoning for combined coordination and reorganization.

3 The AgentCoRe Framework

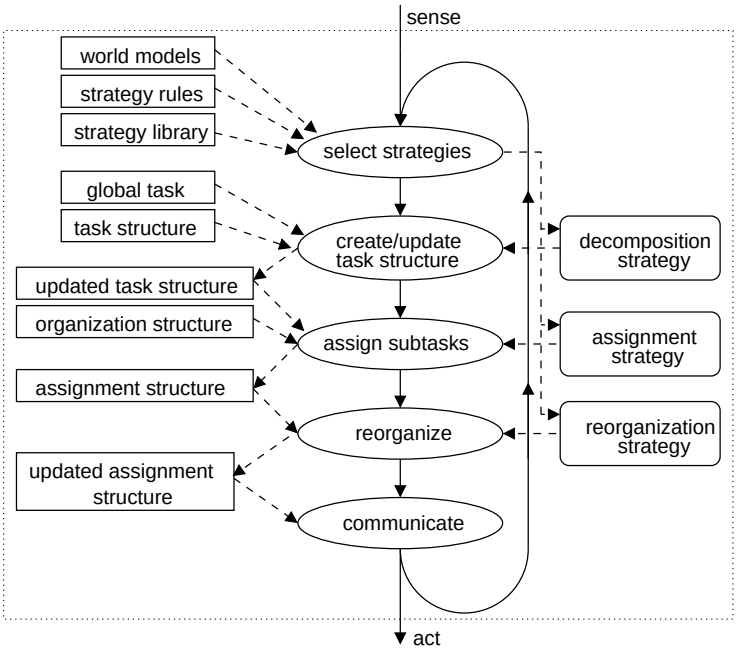


Fig. 1. AgentCoRe.

A global overview of the AgentCoRe framework is shown in figure 1. Oval shapes are sub-processes of the coordination process, rectangles depict declarative knowledge and rounded rectangles depict procedural knowledge. In a single iteration, an agent selects a coordination strategy, decomposes tasks, allocates tasks, reorganizes and communicates.

Strategy selection is based on the current state of the environment and strategy rules which prescribe the use of a coordination strategy in a certain situation. We see a coordination strategy as a combination of a task-decomposition strategy, an assignment strategy and a reorganization strategy. Each of these strategies are used as input for the sub-processes in the coordination process. Strategy selection is an important aspect of our approach because it enables an agent to use different coordination strategies during its lifetime.

The next step in the coordination process is to create and update the task structure. Based on sensory input – which can be observations by the agent and messages from other agents – the agent will decompose the global task into subtasks. The structure of decomposed tasks is called a task structure which is a simplified version of the goal trees in the TÆMS framework [13]. The decomposition strategy describes how the global task decomposes into subtasks.

Relations and interaction patterns between agents and agent roles in the MAS are described in the organization structure. In the task assignment sub-process, subtasks of the task structure are connected to the agents in the organization structure. The task structure combined with the organization structure by means of assignments is called the assignment structure. Which agents are assigned to which tasks is determined by the assignment strategy that is used.

When assignment is completed, the agent can reorganize the assignment structure (which contains the task structure as well as the organization structure). A reorganization strategy describes when and how reorganization takes place. Based on the final assignment structure the agent communicates changes in the organization and task structure to the agents affected by these changes.

3.1 Declarative ingredients

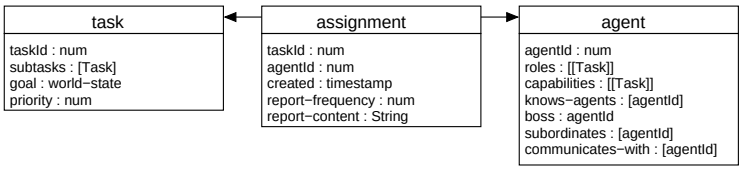


Fig. 2. Basic AgentCoRe declarative concepts.

The basic declarative components of the framework (see figure 2) are `task`, `agent` and `assignment`. A `task` has a set of subtasks and a description of the goal that is to be achieved by performing the task. Furthermore each `task` has a priority. Using the `task` concept, task structures can be created that show how tasks are decomposed into subtasks.

An `agent` has a set of roles the agent is currently enacting (in section 2 a role is defined as a set of tasks) and a set of capabilities which is the set of all roles the agent is capable of enacting. Furthermore, `agent` has relations with other agents. These relations describe which other agents the agent knows, communicates with, is boss of and which agent is its boss. Using the `agent` concept an organization structure can be created where agents have relations with each other, have roles and capabilities.

An `assignment` is a reified relation between `task` and `agent`. It has a timestamp that indicates when the assignment is created, a report frequency that defines when status reports on the progress of the task should be sent, and a specification of the content of the reports. The `assignment` concept connects task structures and organization structures to form assignment structures.

The decomposition, assignment and reorganization strategies are mostly domain specific procedural descriptions of how a specific task should be decomposed or what types of roles should be assigned when reorganizing. Examples of these strategies are given in section 4 of this paper.

3.2 Subprocess description

A knowledge level description of the internal structure of the subprocesses shown in figure 1 is given by using the CommonKADS notation of inference structures [20]. Rectangles depict dynamic information, ovals represent elementary reasoning processes and arrows indicate input-output dependencies. Two thick horizontal lines depict static information used as input for the reasoning processes. For clarity purposes we depict the starting point of the inference structure by a thick squared rectangle.

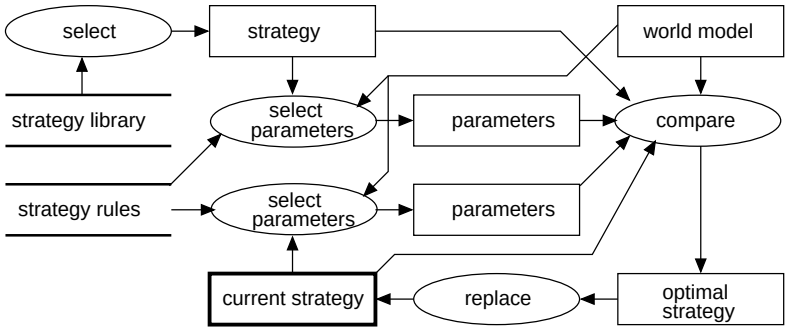


Fig. 3. Strategy selection inference structure.

Figure 3 shows the inference structure of strategy selection in which the current coordination strategy is compared to other available strategies in the strategy library. To obtain the most optimal strategy, parameters are used which represent selection criteria of a coordination strategy (e.g. "required time", "required resources" or "required capabilities"). Strategy rules define in which situation a coordination strategy is optimal by indicating which parameters should be used to compare the strategies. Examples of strategy rules are; "always use the cheapest strategy" and "use the cheapest strategy but when lives are at stake, use the fastest strategy". In the first case, the parameter that indicates cost will be selected. In the second case, the parameters for cost and required time will be selected. The value of a parameter is determined by the current state of the world.

Figure 4 shows the inference structure for task decomposition. First, one of the tasks is selected from the task structure and based on the current model of the world, it is determined whether the task is still valid; is the goal still a valid goal or has a report been received that the task is finished. In the case the task is not valid, the task structure is updated immediately. Otherwise, the task is decomposed as prescribed by the decomposition strategy. The task and the generated subtasks are then used to update the task structure. This continues until each task in the task structure has been validated and decomposed.

In figure 5 the task assignment inference structure (based on the assignment inference structure in [20]) is shown. The assignment strategy determines se-

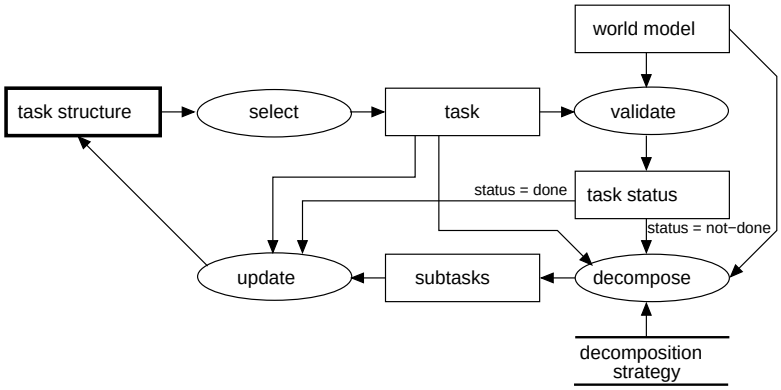


Fig. 4. Task decomposition inference structure.

lection of a set of tasks and a set of agents based on the current assignment structure. Grouping of tasks and agents can be used if multiple agents are assigned to a single task, or one agent is assigned to a group of tasks, or a group of agents is assigned to a group of tasks. If and how grouping is done, depends on the assignment strategy. If no grouping takes place the assign inference will use the task and agent sets that have been selected. The assign inference couples the sets or groups of tasks to the agents which results in a set of new assignments. This continues until all agents are assigned or no tasks are left to perform.

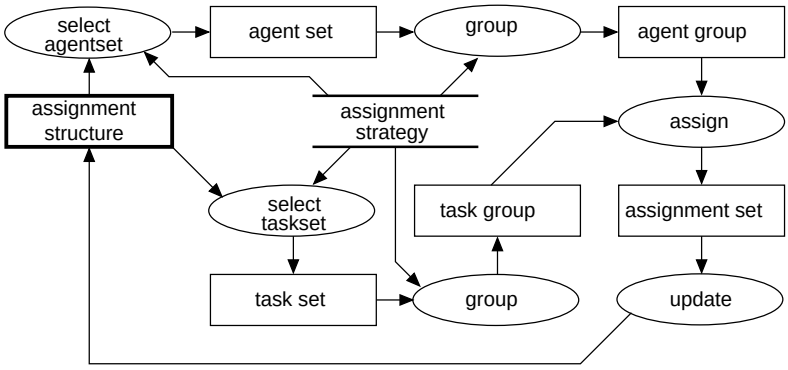


Fig. 5. Task assignment inference structure.

In the reorganize inference structure in figure 6, a reorganization strategy gives a set of available triggers. Triggers are rules that initiate change in organization. Some examples of triggers are detecting an unbalanced workload over the agents, sudden changes in priority of one of the unassigned subtasks while all available agents are already allocated to other tasks, or an event in

the environment that requires two teams to work together. Triggers are tested on the assignment structure and if they fire, a set of matching change rules is selected and applied to the assignment structure. Possible change rules are to assign agents to different roles and creating and/or removing relations between agents, but also taking an agent away from the task it is currently performing and assigning it to a task with a higher priority. Applying change rules results in a partial new assignment structure which is used to update the current assignment structure. Trigger selection continues until all triggers have been tested on the assignment structure.

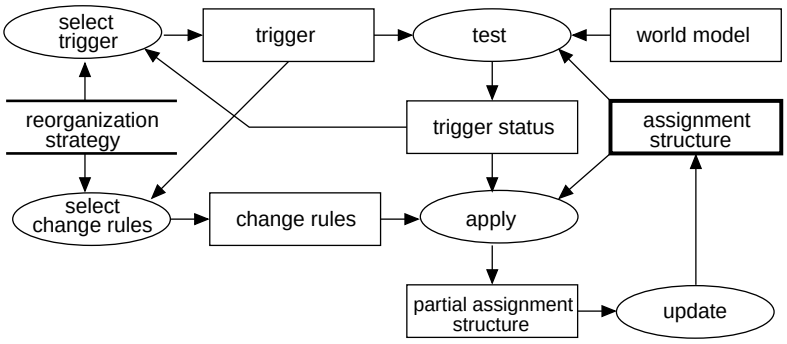


Fig. 6. Reorganize inference structure.

4 A MAS implementation using AgentCoRe

To demonstrate how AgentCoRe is used, the RoboCupRescue simulator [7] is used. In RoboCupRescue, agents are deployed that jointly perform a rescue operation. When the simulation starts, buildings collapse, civilians get injured and buried under the debris, buildings catch fire and fires spread to neighboring buildings. Debris of the collapsed buildings falls on the roads causing roads to be blocked. For this rescue operation, three main tasks can be distinguished and for each of these tasks a type of agent with appropriate capabilities is available. Fires are extinguished by fire brigade teams, blocked roads are cleared by police agents and injured civilians are rescued by ambulance teams.

For the purpose of demonstrating the use of AgentCoRe, we focus on the task of rescuing injured civilians. Thus we have build a MAS that consists of ambulance teams. The tasks for this MAS are the following:

- `SearchAndRescueAll` is the main task of searching the complete map and rescuing all injured civilians. This task has no additional attributes.
- `SearchAndRescueSector`, is the same task as the main task but is restricted to a single sector on the map (the map is divided into 9 sectors). The additional attribute for this task is a `sectorId`.

– `SearchBlock` is the task of searching all houses in a block for injured civilians. Blocks are small groups of houses of which there are 361 on the map. The additional attribute for this task a `blockId`.
– `RescueCivilian` is the task of rescuing a civilian. The additional attributes for this task are a `civilianId` and a `civilianLocation`.
– `CoordinateWork` is the task of coordinating (by means of the AgentCoRe framework) another task. The additional attributes for this task is a `taskId` of the task that is to be coordinated.

Based on the tasks described above, we have defined the following roles in the MAS organization:

– `AmbulanceRole`: [`SearchAndRescueSector`, `SearchBlock`, `RescueCivilian`]
– `GlobalManagerRole`: [`CoordinateWork`]
– `LocalManagerRole`: [`CoordinateWork`, `SearchAndRescueSector`]

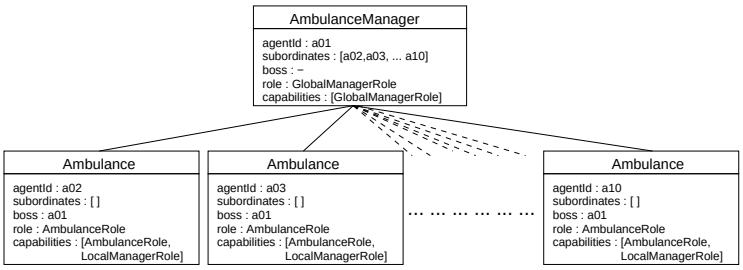
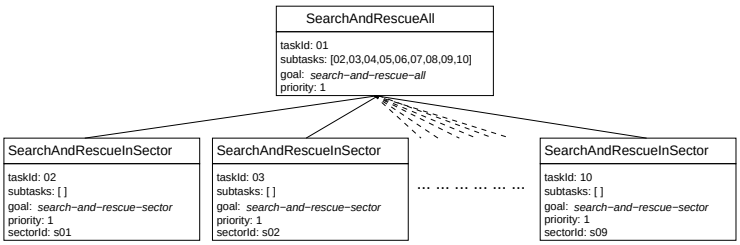


Fig. 7. Ambulance organization, initial structure

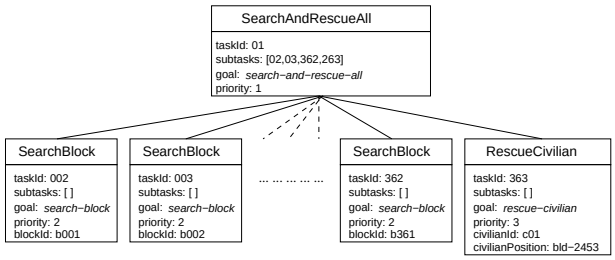
Based on these roles and tasks we have implemented an organization (see figure 7) that is controlled by an `AmbulanceManager` with a `GlobalManagerRole` who coordinates work on the `SearchAndRescueAll` task. The `AmbulanceManager` has 9 `Ambulance` agents at its disposal who are all capable of performing the `AmbulanceRole` and the `LocalManagerRole`. Intially all `Ambulance` agents will perform the `AmbulanceRole`. Because the `communicates-with` and `knows-agent` relations overlap with the authority relations only the authority relations are shown in figures 7 and 9. Authority relations also indicate how tasks are assigned and thus, the `AmbulanceManager` will assign tasks to its direct subordinates and it will receive status reports about the progress of these tasks on a regular basis. An `Ambulance` agent only performs the `LocalManagerRole` when ordered by its direct superior. An agent with the `LocalManagerRole` is able to assign tasks and order role changes to its direct subordinates.

4.1 Strategies

To be able to illustrate the use of different coordination strategies, we have implemented two task decomposition strategies. The first, named decomposition into



(a) decomposition into skills.



(b) decomposition into primitive tasks

Fig. 8. The result of different task decomposition strategies.

skills[1], decomposes the `SearchAndRescueAll` into 9 `SearchAndRescueSector` tasks which results in a task structure as in figure 8(a). The second strategy, named decomposition into primitive tasks, decomposes the `SearchAndRescueAll` into `SearchBlock` tasks for each housing block on the map. If any civilians are reported to be found during one of those `SearchBlock` tasks, a `RescueCivilian` task is generated which results in a task-structure as in figure 8(b). The priority of `RescueCivilian` tasks is based on a civilian's health status. As long as the injury of the civilian is not critical, the priority of the `RescueCivilian` task is lower than the `SearchBlock` tasks. As a civilian becomes more injured, the priority of the `RescueCivilian` task becomes larger than the `SearchBlock` tasks. By adjusting the priority of `RescueCivlian` tasks, the agents search the map as fast as possible but prevent civilians from dying.

For the assignment process we have implemented a strategy that selects tasks from the assignment structure that have not yet been assigned to an agent. From that subset the tasks with the highest priority are selected. The strategy also

[1] The name of this decomposition strategy is based on "coordination by standardization of skills" described by Mintzberg [4]. Coordination by standardization of skills can be characterized by assignment of large and complex tasks to the operator agents.

selects the agents from the assignment structure that are not assigned to a task. The strategy does not include grouping of agents or tasks.

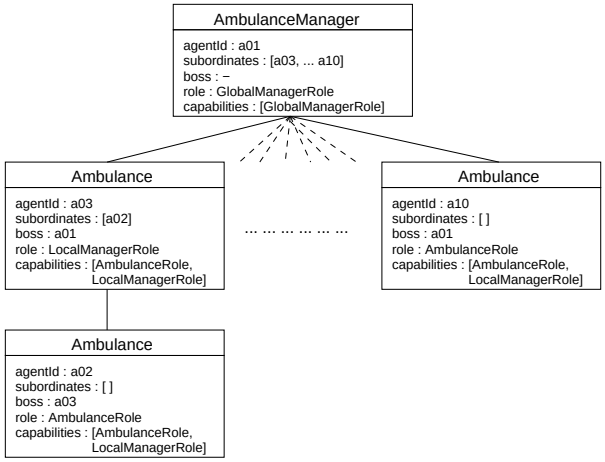


Fig. 9. Ambulance organization, after reorganization

A reorganization strategy has been implemented with one trigger and a set of change rules that are used when the trigger fires. The trigger fires if two conditions both hold; (1) there is at least one agent that has not been assigned to a task and (2) there is at least one task that is still being executed. The change rules specify that the `Ambulance` agent that is already working on that task has to switch from the `AmbulanceRole` to the `LocalManagerRole` (role change) and that the other agent will become a subordinate agent of the agent with the `LocalManagerRole` (structural change). The rationale behind this is that the first agent assigned to a task has acquired the most information on that task and is therefore most suited to coordinate work on this task when other agents are assigned to the same task.

5 Discussion and Conclusions

In the previous section we have described an implementation of a MAS in the RoboCupRescue environment using the AgentCoRe framework. By using a reorganization strategy, the MAS is capable of adapting agent relations and agent roles. Although AgentCoRe allows for switching between coordination mechanisms, the implemented MAS is not capable of changing its coordination style. However, a previous study in [21] has demonstrated the possibility of implementing different coordination mechanisms in an ambulance organization in the RoboCupRescue environment. The coordination mechanisms described in [21] have been composed out of the strategies as described in the previous section.

The design of the AgentCoRe framework enables an agent to use strategies for task-decomposition, task-allocation and reorganization. By using these strategies as input in the inference structures we have been able to distinguish domain dependent strategies from the domain independent reasoning for task decomposition, task assignment and reorganization. By providing the agent with these strategies, the agent will be able to cope with dynamics in the environment [21]. However, it may be the case that the environment or the nature of its task changes in such a way that these strategies – that are designed to enable the agent to cope with these dynamics – are not effective anymore. In this case the agent has the possibility to change its coordination strategy by selecting different strategies for task decompostion, task assignment and reorganization, that are better suited to cope with the current situation.

6 Future Work

As mentioned, the current MAS implementation does not have the capability of adjusting its coordination strategy. We have already shown that AgentCoRe can be used to implement multiple coordination strategies and in future work we will implement strategy rules that allow the agent in a MAS to change its coordination strategy. Future work will also focus on other domains that are more dynamic in nature. Furthermore we will study the applicability of the AgentCoRe framework in these domains to get a better understanding for which types of problem domains AgentCoRe is suited or not.

As also recognized by Dignum et al. [22], different reasons for reorganization exist. Our future research will continue to focus on the questions of *when* a MAS should reorganize and if such a situation occurs, *how* the MAS should reorganize. The first question involves identifying appropriate triggers for strategy selection and reorganization. The second question involves the identification of appropriate change-operators on a MAS organization and determine how these change-operators should be used by the agents in a MAS to achieve a more optimal organization structure.

References

1. So, Y., Durfee, E.: Designing organizations for computational agents. (1998) 47–64
2. Carley, K.: Computational and mathematical organization theory: Perspectives and directions. Journal of Computational and Mathematical Organizational Theory (1995)
3. Newell, A.: The Knowledge Level. Artificial Intelligence **18**(1) (1982) 87–127
4. Mintzberg, H.: Structures in fives: designing effective organizations. Prentice Hall, Englewood Cliffs, N.J. (1993)
5. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Organisational abstractions for the analysis and design of multi-agent systems. In: 1st International Workshop on Agent-Oriented Software Engineering at ICSE 2000. (2000)

6. van Aart, C., Wielinga, B., Schreiber, G.: Organizational Building Blocks for Design of Distributed Intelligent System. International Journal of Human-Computer Studies **61** (2004) 567–599
7. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., Shimada, S.: Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In: Proceedings of IEEE Conference on Man, Systems, and Cybernetics(SMC-99). (1999)
8. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The enterprise ontology. The Knowledge Engineering Review (1998)
9. Carley, K., Gasser, L.: Computational organization theory. In Weiss, G., ed.: Multi-Agent Systems, A Modern Approach to Distributed Artificial Intelligence. MIT-press (1999) 299–330
10. Jennings, N.: Coordination Techniques for Distributed Artificial Intelligence. In: Foundations of Distributed Artificial Intelligence. Wiley (1996) 187–210
11. Tambe, M., Pynadath, D.V., Chauvat, N.: Building dynamic agent organizations in cyberspace. IEEE Internet Computing **4**(2) (2000) 65–73
12. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys **26**(1) (March 1994)
13. Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., Prasad, M.N., Raja, A., Vincent, R., Xuan, P., Zhang, X.Q.: Evolution of the GPGP/TAEMS domain-independent coordination framework. Autonomous Agents and Multi-Agent Systems **9** (2004) 87–143
14. Pynadath, D.V., Tambe, M.: Multiagent teamwork: Analyzing key teamwork theories and models. In: First Autonomous Agents and Multiagent Systems Conference (AAMAS). (2002)
15. Nair, R., Tambe, M., Marsella, S.: Team formation for reformation. In: Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems. (2002)
16. Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organization structures. In: Proceedings of the 5th International Conference on Autonomous Agents, ACM Press (June 2001) 529–536
17. Kamboj, S., Decker, K.: Organizational self-design in semi-dynamic environments. In: 2007 IJCAI workshop on Agent Organizations: Models and Simulations (AOMS@IJCAI 07). (2007)
18. Shehory, O., Sycara, K., Chalasani, P., Jha, S.: Agent cloning: An approach to agent mobility and resource allocation. In: IEEE Communications
19. Barber, K., Martin, C.: Dynamic reorganization of decision-making groups. In: AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, New York, NY, USA, ACM Press (2001) 513–520
20. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press (2000)
21. Ghijsen, M., Jansweijer, W., Wielinga, B.: The effect of task and environment factors on m.a.s. coordination and reorganization. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, short paper, to appear. (2007)
22. Dignum, V., Dignum, F., Sonenberg, L.: Towards dynamic reorganization of agent societies. In: Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004. (September 2004) 22–27

Using Testimonies to Enforce the Behavior of Agents

Fernanda Duran[1], Viviane Torres da Silva[2*], Carlos J. P. de Lucena[1]

[1] Departamento de Informática – PUC, Rio de Janeiro, Brazil
{fduran, lucena}@ inf.puc-rio.br
[2] Departamento de Sistemas Informáticos y Computación – UCM, Madrid, Spain
viviane@fdi.ucm.es

Abstract. Governance copes with the heterogeneity, autonomy and diversity of interests among different agents in multi-agent systems (MAS) by establishing norms. Although norms can be used to regulate dialogical and non-dialogical actions, the majority of governance systems only governs the interaction between agents. Some mechanisms that intend to regulate other agent actions concentrate on messages that are public to the governance system and on actions that are visible by it. But in open MAS with heterogeneous and independently designed agents, there will be private messages that can only be perceived by senders and receivers and execution of actions that can only be noticed by the agents that are executing them or by a group of agents that suffers from their consequences. This paper presents a governance mechanism based on testimonies provided by agents that witness facts that are violating norms. The mechanism points out if agents really violated norms.
Keywords: Open multi-agent system, governance, norms and testimonies.

1 Introduction

Open multi-agent systems are societies in which autonomous, heterogeneous and independently designed entities can work towards similar or different ends [9]. In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, governance (or law enforcement) systems have been defined. Governance systems enforce the behavior of agents by establishing a set of norms that describe actions that agents are prohibited, permitted or obligated to do [3,12]. Such systems assume that norms can sometimes be violated by agents and that the internal state of the agents is neither observable nor controllable.

Different enforcement systems have been proposed in the literature. The majority, such as [10,6], focuses on regulating the interaction between agents. They usually provide governors [6] or law-governed interaction [10] mechanisms that mediate the interaction between agents in order to regulate agent messages and make them comply with the set of norms. Every message that an agent wants to send is analyzed by the mechanism. If the message violates an application norm, the message is not sent to the receiver. The main disadvantages of such approaches are (i) they influence

* Research supported by the Juan de la Cierva program, Comunidad de Madrid Program S-0505/TIC-407 and by Spanish MEC Projects TIC2003-01000.

the agents' privacy since those mechanisms interfere in every interaction between agents and (ii) they do not govern non-dialogical actions since they only concern about the compliance of messages with the system norm [13]. Non-dialogical actions are related to tasks executed by agents that characterize, for instance, the access to resources, their commitment to play roles or their movement in environments and organizations.

Other approaches provide support for the enforcement of norms that regulate not only the interactions between agents but also the access to resources [4] and the execution of agent's actions [13]. TuCSoN [4] provides a coordination mechanism to manage the interaction between agents and also an access control mechanism to handle communication events, in other words, to control the access to resources. In TuCSoN agents interact through a multiplicity of independent coordination media, called tuple centres. The access control mechanism controls agent access to resources by making the tuple centres visible or invisible to them. Although in TuCSoN norms can be described to govern the access to resources, the governance is restricted and only applied to resources that are inserted in tuple centre environments.

In [13] the authors claim that the governance system enforces the observable behavior of agents in terms of public messages and visible actions. They introduce a classification of norms and, according to such classification, they provide some implementation guidelines to enforce them. The main drawback of this approach is that it does not provide support for the enforcement of messages and actions that are not directly accessed by the governance system. Such an approach assumes that the governance system can enforce every norm since it can access all messages and actions regulated by a norm. But in open MAS with heterogeneous and independently designed agents, there will be private messages that can only be perceived by senders and receivers and execution of actions that can only be noticed by the agents that are executing them or by a group of agents that suffers from their consequences [1].

In this paper we propose a governance mechanism based on testimonies provided by witnesses about facts or events that they know are related to norm violations. Agents are inserted in an environment where they can perceive the changes occurred in it. Since agents can observe these changes, they can provide testimonies about actions or messages that are in violation of a norm. In our approach, private messages and also private actions can be enforced. *Private messages* that violate norms can be testified by agents that are involved in the interactions. Such agents can testify about messages they should have received or about messages they should have not received. *Private actions* that are executed in the *scope of a group* and are violating norms can be testified by any member of the group that knows such norms and has seen the actions being executed or has perceived facts or events that reflect the execution of such actions. The same can be said about actions that should have been executed but were not. Related facts or events cannot be observed and, therefore, agents can testify stating that the actions (probably) were not executed. In addition, *private actions* that are executed in the *scope of one single agent* and that are violating norms can be testified by any agent that knows the norms and that perceives facts or events that are related to the execution of such actions. The same can be said about actions that an agent should have executed but has not. Other agents that know the norms that regulate such actions can testify if they cannot observe the related facts or events.

The paper presents in Section 2 an overall view of the testimony-based governance mechanism. Section 3 details the judgment process used by the mechanism while Section 4 describes a case study where we apply our approach. Finally, section 5 concludes and describes some advantages and drawbacks of our proposal.

2 The Testimony-Based Governance Mechanism

The governance mechanism presented here is based on testimonies that agents provide attesting facts or events that may be norm violations. Since every agent knows sets of norms, it can report to the governance mechanism their violation.

2.1 Governance Mechanism Assumptions

The testimony-based governance mechanism is funded in the following assumptions.

Assumption I: *Every agent should know every norm applied to itself.* Such as in the real world where everyone should know a code of behavior, we assume that every agent should know all norms that can be applied to their messages or actions independently of the system environment in which it is executing. When an agent enters in the environment to play a role, the environment/system must be able to provide to the agent all norms applied to this role. This is important because the mechanism assume that an agent acting in violation of a norm chooses to do so being aware of that. The set of norms that regulates the application should by provided by an ontology.

Assumption II: *Every agent should know every norm that influences its behavior and should be able to observe violations of such norms.* Agents should know the norms that regulate the behavior of other agents when the violations of such norms influence their own execution. Therefore, when entering in an environment, agents should not only observe the norms applied to the roles they will play, but also the norms that, when violated by other agents, influence their execution.. The possible violation of such norms motivates the agents to be aware of them.

Assumption III: *Every agent can give testimonies about norm violations.* Since an agent knows norms that are applied to other agents, the agent should be able to state that one of these norms is being violated. Every time an agent perceives the violation of a norm, it must be able to give a testimony to the governance mechanism. The proposed mechanism provides a component that can be used by agents to help them analyzing their beliefs in order to find out well-known facts or events that may be norms violations.

Assumption IV: *Some violations might be ignored / not observed.* The proposed mechanism does not impose that an agent must give its testimony whenever it notices a norm violation. Agents should be well motivated in order to provide their testimonies. Besides, the mechanism does not guarantee that all violations will be observed by at least one agent. It may be the case that a violation occurs and no agent testifies about it.

Assumption V: *Agents can give false testimony.* In an open system, agents are independently implemented, i.e. the development is done without a centralized control

and the governance mechanism cannot assume that an agent was properly designed. Therefore, there is no way to guarantee that all testimonies are related to actual violations. So, the governance mechanism should be able to check and assert the truthfulness of the testimonies.

Assumption VI: *The mechanism can have a law-enforcement agent force.* The mechanism can introduce agents which have the sole purpose of giving testimonies. The testimonies of those agents provided by the mechanism can always be considered to be truthful and the judgment subsystem can directly state that a norm was violated and a penalty should be assigned. Note that those agents must only testify if they are sure about the culpability of the application agents and that they can only testify about violations related to public messages and actions. They must be aware that an agent may violate a norm due some major force or to another agent fault, for instance.

2.2 The Governance Mechanism Architecture

In order to decentralize the governance of large-scale multi-agent systems, we propose to use a hierarchy of organizations where agents are executing according to their roles. Each system organization should state its own norms and implement the proposed governance system to regulate them. The mechanism's architecture proposes three subsystems. The judgment subsystem is responsible for receiving the testimonies and for providing a decision (or verdict) pointing out to the reputation and sanction subsystems if an agent has really violated a norm. The system may use different strategies to judge the violation of the different norms specified by the application. Such strategies might use the agents' reputation afforded by the reputation system to help providing the decision. It is well established that trust and reputation are important in open systems and can be used by the agents for reasoning about the reliability of other ones [11]. In [11] trust is defined as subjective probability with which agents assess that other agents will perform a particular action. We adapt this definition to our approach stating that reputation is defined as a subjective probability with which agents assess that other agent will provide trustful testimonies. The reputation subsystem [7] evaluates the reputation of agents according to the decisions provided by the judgment subsystem about violated norms and false testimonies. Finally, the third subsystem, the sanction subsystem, applies the sanctions specified in norms to the witness agents or to the defendant agents, according to the judgment decision.

3 The Judgment Sub-system

The judgment sub-system has three main responsibilities: to receive testimonies, to judge them and to provide the decision about the violation. Three different agent types were defined to deal with these responsibilities: inspector, judge and broker agents. The inspector agents are responsible for receiving the testimonies and sending them to judge agents. The judge agents examine the testimonies and provide decisions that are sent to broker agents. Broker agents are responsible for interacting with the reputation and sanction sub-systems to make the decisions effective. While judging

the testimonies, judge agents may interact with brokers to get information about the reputation of agents.

3.1 The Judgment Process

The judgment process is composed of eight steps where six are application independent ones. Although judgment strategies cannot be completely independent of the application norms, it is possible to define some common steps to be followed by any judgment strategy. In this section we present the eight steps that compose the judgment process.

Step I: *To check if the testimony has already been judged.* Agents may send testimonies about facts that have already been testified and judged. Because of that, the first step of the judgment process checks if the testimony is related to one of the judgment processes that had occurred before and had considered the defendant guilty. If so, the testimony is discarded and the judgment process is canceled.

Step II: *To verify who the witness is.* According to assumption VI, the testimony provided by some specific agents must be considered always truth. Therefore, the second step of the judgment process verifies who the witness is. If it is the case of an always truthful witness, the judgment process is finished and the verdict stating that the agent must be penalized is provided.

Step III: *To check if the norm applies to the defendant agent.* According to assumption V, agents can lie and end up accusing other agents of violating norms that are not applied to them. In order to find out if a testimony is true, this step checks if the norm applies to the defendant agent. If the norm does not apply, the judgment process is finished and the verdict states that the defendant agent is absolved.

Step IV: *To ask the defendant agent if it is guilty.* If the norm applies to the agent, the next step is to ask it if it has violated the norm it is accused of. As it happens in the real world, if the agent confesses, the judgment process is finished and the verdict states that the defendant agent is condemned. Otherwise, the judgment process continues. In cases where the defendant confesses the violation, the applied punishment can be smaller than the one that would be applied if he hasn't confessed.

Step V: *To judge the testimony according to the norm (application dependent step).* If the agent did not confess, it is necessary to carefully examine if the agent really violated the norm. In order to determine if the testimony is truth and, therefore, if the defendant agent is guilty, it may be necessary to use different strategies for different violated norms. For instance, on one hand, if the norm regulates the payment of an item and the defendant is being accused of having not paid the witness, one possible strategy is to ask the defendant if it has the receipt signed by the witness asserting that it has received the payment. On the other hand, if the norm states that an agent should have not updated a resource, the judgment system could use the simple strategy that checks the resource log, in case it is provided. It is clear that such strategies are application dependent ones since they depend on the norm that is being enforced.

Step VI: *To ask other agents about their depositions (application dependent step).* If the application strategy could not decide if the defendant agent is guilty or not, the judgment system can still try another approach. Since there may be other agents that

can also testify about the violation of the norm or facts related to it, the judgment system can explicitly ask them about their opinion about the violation. This step is an application dependent step because depending on the kind of question the judgment system makes to the agents, it may be necessary to interpret the answer according to the application norm being checked. For instance, two different kinds of questions can be asked to those agents: (i) Have you seen agent a_i violating norm n_j? (ii) What do you know about fact f_k? There are different interpretations for each of the questions and such interpretations are application dependent.

Step VII: *To come up with a consensus considering the depositions.* After interpreting the depositions, the judgment system must put them together to come up with a verdict. In order to do so, our approach uses the agent reputations to help evaluating the depositions. The consensus between the depositions is provided by using subjective logic [8], as detailed in Section 3.3. Such an approach evaluates the depositions considering the reputations of the agents to come up with the probability of the defendant agent being guilty of violating the norm.

Step VIII: *To provide the decision.* The judgment system can provide three decisions. It can state that (i) the defendant agent is probably guilty, (ii) the defendant is probably not guilty (the witness has lied), or (iii) the culpability of the defendant is undefined. In this case, the judge could not decide if the agent is guilty or not.

After producing the decision, it is necessary to send it to the reputation sub-system so that it can modify the reputation of the accused agent, in case the judgment system has decided that the defendant agent is guilty, or the reputation of the witness, in case the judgment system has decided that it has lied. It is also important to inform the decision to the sanction sub-system to (i) punish the agent for violating a norm and to award the witness for providing the testimony or (ii) to punish the witness for providing an untruthful testimony.

3.2 Evaluating the Testimonies and Depositions

When there are not enough evidences to be used by the judge agent to come up with a decision, it can still make use of agents' depositions to finally provide a verdict, as described in Step VI and VII. However, as stated before in assumption V, agents can give false testimonies and also false depositions. Therefore, there is a need for an approach that evaluates such testimonies and depositions considering the reliability of the agents, i.e., considering their reputations. We propose the use of subjective logic to provide a verdict stating the probability of an agent being guilty or not for violating a norm. Such an approach is used in the application independent Step VII to ponder the testimonies/depositions according to the agents' reputations and to make a consensus between them.

In [5] the authors sketched a model for e-marketplaces based on subjective logic for setting contracts back on course whenever their fulfillment deviate from what were established. Evidences from various sources are weighed in order to inform the actions that are probably violating the contracts. Subjective logic is used to support reasoning over those evidences, which involve levels of trust over parties, combining recommendations and forming consensus.

In [2], to evaluate the trustworthiness of a given party, especially prior to any frequent direct interaction, agents may rely on other agents (witnesses) who have interacted with the party of interest. The testimonies given by those witnesses are based on direct interactions and may hold a degree of uncertainty. To combine the testimonies and create a single opinion (reputation) about an agent, the authors used the Dempster-Shafer theory of evidence as the underlying computational framework.

3.2.1 Introducing Subjective Logic

Subjective Logic was proposed by Audun Jøsang based on the Dempster-Shafer theory of evidence [8]. This approach addresses the problem of forming a measurable belief about the truth or falsity on an atomic proposition, in the presence of uncertainty. It translates our imperfect knowledge about reality into degrees of belief or disbelief as well as uncertainty which fills the void in the absence of both belief and disbelief [8]. This approach is described as a logic which operates on subjective beliefs and uses the term *opinion* to denote the representation of a subjective belief. The elements that compose the frame of discernment which is a set of all possible situations are described as follows: (i) The agent's opinion is represented by a triple $w(x) = <b(x), d(x), u(x)>$; (ii) $b(x)$ measures belief, represented as a subjective probability of proposition x to be true; (iii) $d(x)$ measures disbelief, represented as a subjective probability of proposition x to be false; (iv) $u(x)$ measures uncertainty, represented as a subjective probability that a proposition x to be either true or false; (v) $b(x), d(x), u(x) \in [0..1]$; $b(x) + d(x) + u(x) = 1$; and (vi) $w^A(x)$ represents the opinion that an agent A has about the proposition x to be true or false.

Subjective Logic operates on opinions about binary propositions, i.e. opinions about propositions that are assumed to be either true or false. The operators described above are to be applied over such opinions.

Recommendation (Discounting): The *discounting* operator $\otimes$ combines agent A's opinion about agent B's advice with agent B's opinion about a proposition x expressed as an advice from agent B to agent A. That means if agent B gives an advice x to agent A, and agent A has an opinion about agent B, the operator $\otimes$ can be used to form *agent A's opinion about agent B's advice x*: (i) $w^A(B) = <b^A(b),d^A(b),u^A(b)>$ represents agent A's opinion about agent B; (ii) $w^B(x)=<b^B(x),d^B(x),u^B(x)>$ represents agent B's opinion about x; (iii) $w^{A:B}(x)= w^A(B) \otimes w^B(x)$ represents agent A's opinion about agent B's opinion about the preposition x. $w^{A:B}(x)=<b^{A:B}(x),d^{A:B}(x),u^{A:B}(x)>$ and is evaluated as follows:

$b^{A:B}(x) = b^A(b)\, b^B(x)$; $d^{A:B}(x) = b^A(b)\, d^B(x)$; $u^{A:B}(x) = d^A(b) + u^A(b) + b^A(b)\, u^B(x)$

Consensus: The *consensus* of two possibly conflicting opinions is an opinion that reflects both opinions in a fair and equal way, i.e. when two observers have beliefs about the truth of x, the consensus operator $\oplus$ produces a consensus beliefs that *combines the two separate beliefs into one*: (i) $w^A(x) = <b^A(x),d^A(x),u^A(x)>$ represents agent A's opinion about x; (ii) $w^B(x) = <b^B(x),d^B(x),u^B(x)>$ represents agent B's opinion about x; (iii) $k = u^A(x) + u^B(x) - u^A(x)u^B(x)$; and (iv) $w^{A,B}(x) = w^A(B) \oplus w^B(x)$ represents the consensus between agent A's opinion about x and agent B's opinion about x. $w^{A,B}(x)= <b^{A,B}(x),d^{A,B}(x),u^{A,B}(x)>$ is calculated as follows for $k \neq 0$:

$b^{A,B}(x)=(b^A(x)u^B(x)+b^B(x)u^A(x))/k$; $d^{A,B}(x)=(d^A(x)u^B(x)+d^B(x)u^A(x))/k$;

$u^{A,B}(x)=(u^A(x)\, u^B(x)) / k$

3.2.2 Applying Subjective Logic in our approach

Our goal is to come up with a consensus between the different testimonies and depositions about the violation of a norm considering the reliability of the witnesses. In order to do so, it is important to understand what a testimony/deposition is in the context of subjective logic. The testimony or deposition given by agent A attesting something about a proposition x can be seen as the *A's opinion about x, i.e.,* $w^A(x)$.

Second, it is necessary to state that the testimonies (or the opinions of the agents about facts) will be evaluated by the judge agent according to its own opinion about the agents, for instance, $w^J(a)$ where A is one of the witnesses. Such an opinion is directly influenced by the reputation of the agent.

After evaluating the judge's opinions about the agents that have given their testimonies and depositions, it is necessary to evaluate the judge's opinions about testimonies and depositions given by those agents. In order to do so the *discounting operator* will be used. Finally, after having the judge's opinions about all testimonies and depositions, it is necessary to put them all together to form the judge point of view about the violated norm. The *consensus operator* is therefore used.

Judge's opinions about the agents:

The reputation provided by the reputation system reflects how much the judge believes in the agent, i.e. $b^J(a)$, and not its whole opinion about such agent, i.e $w^J(a)$.

Judge's opinions about testimonies and depositions given by the agents:

The judge's opinion about a testimony/deposition given by an agent, i.e $w^{J:A}(x)$, depends on the judge's opinion about the agent, $w^J(a)$, and the agent's opinion about fact x that is related to the testimony/deposition, $w^A(x)$. In order to evaluate the judge's opinion we use the discounting operator presented in Section 3.2.1 as described in equation (1): $w^{J:A}(x) = w^J(a) \otimes w^A(x) = < b^{J:A}(x), d^{J:A}(x), u^{J:A}(x)>$ (1)

Judge's point of view about the violated norm:

Given that there may exist more then one agent testifying about the same fact (proposition x), all testimonies and depositions can be combined using the consensus operator to produce the judge's own opinion about the proposition x. The consensus puts together all testimonies and depositions while considering the reputation of the witnesses. For instance, let's suppose that A, B and C are agents that provided their testimonies and depositions, the consensus is formed by using equation (2):

$$w^{J:(A,B,C)}(x) = (w^J(a) \otimes w^A(x)) \oplus (w^J(b) \otimes w^B(x)) \oplus (w^J(c) \otimes w^C(x)) \quad (2)$$

4 A Case Study: Cargo Consolidation and Transportation

In order to validate our approach we present a case study based on the real-life cargo consolidation and transportation domain. Cargo consolidation is the act of grouping together small shipments of goods (often from different shippers) into a larger unique unit that is sent to a single destination point (and often to different consignees). Such practice makes possible to the enterprises that provide transportations to reduce the rate of shipping. Importers and exporters that want to ship small cargos may look for consolidator's enterprises that provide cargo consolidation to ship their goods.

An open multi-agent system approach is entirely adequate for developing applications on this domain because such applications mostly involve interactions

between different autonomous partners playing different roles in order to accomplish similar objectives. Such applications are governed by several rules that are used to regulate the behavior of the heterogeneous and independently designed entities that reinforce the open characteristic of the systems. In this paper we will contemplate examples of two different norms that are regulated by the proposed mechanism.
Norm I: *The consolidator agent must not change its shipment schedule once it has been presented.*
Norm II: *The consolidator agent must deliver the cargo at the destination on the date established in the transportation agreement.*

4.1 Norm I

In this section we present the judgment process that judge testimonies stating that norm I was violated. We detail the two application dependent steps (Steps V and VI) and also the application independent Step VII that makes a consensus between the testimonies. Let's suppose that a testimony was provided by one of the application agents (an importer, for instance) stating that the agent consolidator has violated norm I. After checking that the testimony is not about a fact that has already been judged (Step I), that the witness is not a law-enforcement agent (Step II), that norm I really applies to the defendant agent (Step III) and that the defendant did not confess that it has violated the norm (Step IV), it is necessary to judge the testimony according to the particular characteristics of norm I (application dependent Step V).

In order to judge testimonies stating violation of norm I, such testimonies must inform shipment schedule firstly defined by the consolidator agent and the actual shipment schedule. One possible application strategy to judge such testimonies is described below. It supposes that there is a system's resource that stores the shipment schedules. The resource is analyzed with the aim to compare the information provided in the testimony with the stored information. If the schedule provided by the resource is equal to the first schedule available in the testimony, the schedule was not changed and the testimony is discarded. If the schedule provided by the resource is different to the actual schedule provided by the testimony, the testimony is also discarded because the testimony describes a fact that cannot be confirmed. In both cases the witness is providing a false testimony. The judgment process is finished and the defendant is considered 100% innocent (Step VIII).

Nevertheless, if the schedule provided by the resource is equal to the actual schedule provided by the testimony, the judgment process should continues in order to find out if the schedule was really changed. Since the application does not have logs to inform when resources are updated, the alternative to find out if the consolidator agent has really changed the schedule is to ask other agents about their opinions (application dependent Step VI). The information provided by the witness is confronted with the information provided by other agents, in this case, with the opinion of two others importers and two exporters about the violation of norm I.

The decision (Step VII) is established based on the information provided by the testimony, the defendant statement and the importers' and exporters' depositions by using subjective logic. Such testimonies and depositions are analyzed from the point of view of the judge and, therefore, there is a need for evaluating how much the judge

believes in each agent. As stated before, the reputation of the agent (provided by the reputation system) reflects how much the judge believes in the agent; $b^J(a) = rep(a)$.

The judge's beliefs are used to evaluate the judge's opinion about the testimonies and depositions provided by the agents. Such opinions ($w^{J:W}(x)$, $w^{J:C}(x)$, $w^{J:I1}(x)$, $w^{J:I2}(x)$, $w^{J:E1}(x)$ and $w^{J:E2}(x)$), evaluated by using equation (2), are depicted in Table 1. We are supposing that the two importers and the two exporters, together with the witness, have stated that the defendant is guilty ($w^A(x)$).

The verdict, i.e the judge point of view about the violated norm, can be provided by applying the consensus operator (equation (2)). In this example the verdict (equation (3)) states that the probability of the consolidator agent has violated norm I is 84%.

$$w^J = w^{J:W}(x) \oplus w^{J:C}(x) \oplus w^{J:I1}(x) \oplus w^{J:I2}(x) \oplus w^{J:E1}(x) \oplus w^{J:E2}(x) = \mathbf{<0.84}, 0.06, 0.1> \quad (3)$$

Table 1. Judge's opinion about the violation of norm I

	Statement	$\mathbf{w^A(x)}$	$\mathbf{b^J(a)}$	$\mathbf{w^J(a) \otimes w^A(x) = w^{J:A}(x)}$
Witness	Guilty	<1,0,0>	0.54	$w^{J:W}(x) = <0.54,0,0.46>$
Consolidator Agent	Innocent	<0,1,0>	0.33	$w^{J:C}(x) = <0,0.33,0.67>$
Importer1	Guilty	<1,0,0>	0.75	$w^{J:I1}(x) = <0.75,0,0.25>$
Importer2	Guilty	<1,0,0>	0.53	$w^{J:I2}(x) = <0.53,0,0.47>$
Exporter1	Guilty	<1,0,0>	0.57	$w^{J:E1}(x) = <0.57,0,0.43>$
Exporter2	Guilty	<1,0,0>	0.66	$w^{J:E2}(x) = <0.64,0,0.34>$

4.2 Norm II

In this section we also focus on the two application dependent steps (Steps V and VI) and on Step VII while illustrating the judgment process of norm II. As in Section 4.1, we assume that the judge system could not provide a verdict before executing Step V.

In order to judge testimonies stating violations of norm II, such testimonies must contain the transportation documents called House Bill of Landing (HBL) and Master Bill of Landing (MBL). A bill of landing is a document issued by the carrier (the consolidator agent, in this case) that describes the goods, the details of the intended transportation, and the conditions of the transportation. The difference between HBL and MBL is that the MBL describes several small cargos consolidated in a single shipment and the HBL describes each small cargo.

Therefore, in step V, the judge must first ensure that the exporter has really delivered the cargo at the place designated by the consolidator on the appropriated date. When this task is accomplished, the consolidator gives a copy of the HBL (related to the cargo delivered by the exporter) to the exporter. The judge can, therefore, ask the exporter about his copy of the HBL. If the exporter does not have this document, the judgment process is finished, the witness' testimony is considered false and the defendant is considered 100% innocent (Step VIII). The consolidator agent has not delivered the cargo because the exporter has not delivered its cargo to the consolidator agent.

On the other hand, if the exporter has its copy of the HBL the judge must execute step VI, continuing the judgment process to come to a verdict. Since, the witness'

cargo has been consolidated with others cargos, the judge may ask all other importers mentioned in the MBL if their cargos have been delivered in the correct date and place. After receiving the importers depositions, the judge needs to execute step VII, where it puts together all statements while considering the reputations of consolidator agent and all importers of the mentioned shipment. We are supposing that there were three cargos consolidated in this shipment. Table 2 depicts the judge's opinion about the testimony and depositions provided by the witness, the consolidator agent and the two importers ($w^{J:C}(x)$, $w^{J:I1}(x)$, $w^{J:I2}(x)$ and $w^{J:I3}(x)$).

The verdict, i.e judge point of view about the violated norm, can be provided by applying the consensus operator, as shown in equation (4). In this example the verdict states that the probability of the consolidator agent has violated norm II is 76%.

$$w^{J} = w^{J:W}(x) \oplus w^{J:C}(x) \oplus w^{J:I1}(x) \oplus w^{J:I2}(x) = \mathbf{<0.76}, 0.18, 0.06> \quad (4)$$

Table 2. Judge's opinion about the violation of norm II

	Statement	$\mathbf{w^{A}(x)}$	$\mathbf{b^{J}(a)}$	$\mathbf{w^{J}(a) \otimes w^{A}(x) = w^{J:A}(x)}$
Witness	Innocent	<0,1,0>	0.75	$w^{J:W}(x)$ = <0,0.75,0.25>
Consolidator Agent	Guilty	<1,0,0>	0.23	$w^{J:C}(x)$ = <0.23,0,0.77>
Importer1	Guilty	<1,0,0>	0.47	$w^{J:I1}(x)$ = <0.47,0,0.53>
Importer2	Guilty	<1,0,0>	0.92	$w^{J:I2}(x)$ = <0.92,0,0.08>

The approaches that governs only the interactions between agents, such as [10,6], could not govern norm I since this norm govern the access to a resource. As stated in Section 1, there are approaches that govern the public messages and visible actions, both in the system point of view. Such approaches could only be used to enforce norm I and II if we consider (i) that the shipment schedules of a consolidator agent are public resources and, therefore, every action done in such resource are visible actions and (ii) that the deliveries done by the consolidator agent are public messages, that is not usually the case. Moreover, note that both strategies presented in section 4.1 and 4.2 are simple examples that can be used to judge the testimonies related to norms I and II. Other more complex and completely different strategies could have been implemented to judge the same testimonies.

5 Conclusion

In this paper we present a governance mechanism based on testimonies given by agents that have perceived norm violations. Since a violation of a norm influences (injures) the execution of an agent, perceiving it will be a natural consequence of the regular execution of that agent. The mechanism judges the testimonies it receives trying to differentiate true and false testimonies in order to provide a verdict. [(1)]The governance mechanism was implemented as a framework that supports, by now, the judgment and reputation sub-systems (section 2.2). The main advantages of the proposed mechanism are: (i) it does not interfere in the agents' privacy; (ii) it can be used to enforce norms associated not only with interactions but also with the execution of different actions, such as the access to resources; and (iii) it does not

assume that the system can do all the work of finding out the violations and enforcing the norms.

Whereas we believe that the advantages of our proposed mechanism are really important, it has some potential weaknesses. First, it may be difficult to distinguish if a testimony is true or false and, therefore, to provide a good verdict. We proposed to solve this problem by using probability based on subjective logic while providing the verdicts. Second, violations that go without testimonies will not be punished. This could lead to an undesired system state. One way to overcome this issue is motivating the agents to give their testimonies by using an agent rewards program, for instance. Another important drawback is that the effort to implement an agent under the proposed governance system may increase since it needs not only to perceive facts, but also to associate them with possible norm violations. To minimize this impact, the judgment subsystem provides a mechanism that can be used by the agents to associate facts with norms violations. In order to improve our work we are in the way of adding some argumentation aspects to the judgment process. This will improve the set of evidences used for and against a verdict.

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 Proc. of the Workshop on Coordination, Organization, Institutions and Norms in agent systems (2006) 223-225.
2. Bin Yu, Singh, M.: Detecting Deception in Reputation Management. In Proc. of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003) 73 – 80.
3. .Boella, G.; van der Torre, L.: Regulative and Constitutive Norms in Normative Multi-Agent Systems. In Proceeding of KS. AAAI Press, (2004) 255-265.
4. Cremonini, M., Omicini, A., Zambonelli, F.: Coordination and Access Control in Open Distributed Agent Systems: The TuCSoN Approach. LNCS 1906, (2000) pp 99-114.
5. Daskalopulu, A., Dimitrakos T., Maibaum T.: E-Contract Fulfilment and Agents' Attitudes. In Proc. ERCIM WG E-Commerce Workshop on The Role of Trust in e-Business, (2001).
6. Esteva, M., de la Cruz, D., Sierra, C.: Islander: An Electronic Institutions Editor. In Proc. of Int. Conf. on Autonomous Agents and Multi-Agent Systems (2002) 1045–1052.
7. Guedes, J., Silva, V., Lucena, C. J. P.: A Reputation Model Based on Testimonies. In Proceedings of Workshop on Agent-Oriented Information Systems at CAiSE (2006) 37-47.
8. Jøsang A.: An Algebra for Assessing Trust in Certification Chains. In Proc. Network and Distributed Systems Security Symposium (1999).
9. López, F.: Social Powers and Norms: Impact on Agent Behaviour. PhD thesis. University of Southampton. UK (2003)
10. Minsky, N., Ungureanu, V.: Law-Governed Interaction: A Coordination & Control Mechanism for Heterogeneous Distributed Systems. ACM TSEM, July,9(3)(2000)273-305.
11. Patel, J., Teacy, W., Jennings, et al.: Monitoring, Policing and Trust for Grid-Based Virtual Organizations. In Proc. of the UK e-Science All Hands Meeting 2005 UK (2005) 891-898.
12. Singh, M.: An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law v. 7 (1), Springer (1999) 97-113.
13. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Implementing Norms in Multiagent Systems. LNAI 3187. Springer-Verlag (2004) 313 - 327

Implementing Norms that Govern Non-Dialogical Actions

Viviane Torres da Silva*

Departamento de Sistemas Informáticos y Computación – UCM, Spain, Madrid
viviane@fdi.ucm.es

Abstract. The governance of open multi-agent systems is particular important since those systems are composed of heterogeneous, autonomous and independently designed agents. Such governance is usually provided by the establishment of norms that regulate the actions of agents. Although there are several approaches that formally describe norms, there are still few of them that propose their implementation. In this paper we propose the implementation of norms that govern non-dialogical actions by extending one of the approaches that regulate dialogical ones. Non-dialogical actions are not related to the interactions between agents but to tasks executed by agents that characterize, for instance, the access to resources, their commitment to play roles or their movement into environments and organizations.
Keywords: norm, governance of multi-agent system, non-dialogical action, implementation of norm

1 Introduction

The governance of open multi-agent systems (MAS) copes with the heterogeneity, autonomy and diversity of interests among agents that can work towards similar or different ends [9] by establishing norms. The set of system norms defines actions that agents are prohibited, permitted or obligated to do [1,12].

Several works have been proposed in order to define the theoretical aspects of norms [3,5], to formally define those norms [2,4], and to implement them [7,8,9,10,13]. In this paper we focus on the implementation of norms. Our goal is to present an approach where dialogical and non-dialogical norms can be described and regulated. Non-dialogical actions are not related to the interactions between agents but to tasks executed by agents that characterize, for instance, the access to resources, their commitment to play roles or their movement in environments and organizations. From the set of analyzed proposals for implementing norms, few approaches considers non-dialogical actions [9,10,13]. Although, the authors present some issues on the verification and enforcement of norms, they do no demonstrate how such issues should be implemented. Other approaches such as [7,8] deal with e-Institutions and, thus, consider illocutions as the only action performed in such systems.

Our approach extends the work presented in [8] with the notion of non-dialogical

* Research supported by the Juan de la Cierva programa, Comunidad de Madrid S-0505/TIC-407 and MEC-SP TIC2003-01000.

actions proposed in [13]. A normative language is presented in [8] to describe illocutions (dialogical actions) that might be dependent on temporal constraints or the occurrence of events. We have extended the normative language in order to be possible to specify non-dialogical norms that state obligations, permissions or prohibition over the execution of actions of agents' plans (as proposed in [13]) and of object methods. Similar to the approach presented in [8], we have also used Jess[1] to implement the governance mechanism that regulates the behavior of agents. The mechanism activates norms and fires violations (Jess rules) according to the executed (dialogical or non-dialogical) actions (Jess facts).

The paper is organized as follows. Section 2 describes the example we are using to illustrate our approach. Section 3 intends to clearly present the difference between dialogical and non-dialogical actions. Section 4 points out the main concepts of the extended normative language and Section 5 describes the implementation of the governance engine in Jess. Section 6 concludes our work.

2 Applied Example

In order to exemplify our approach, we have defined a set of six norms that govern a simplified version of a soccer game. The soccer game is composed of agents playing one of the three available roles: referee, coach and player (kicker or goalkeeper). The responsibilities of a referee in a soccer game are: to start the game, stop it, check the players' equipments and punish the players. The available punishments are: to show a yellow card, send off a player, and declare a penalty. The possible actions of a player during a game are: kick the ball and handle the ball. The coach role is limited to substitute players. Besides those actions, all agents are able to move and, therefore, enter and leave the game field. The six norms that regulate our simple soccer game are the following:

Norm 1: *The referee must check the players' equipments before starting the game.*
Norm 2: *A coach cannot substitute more than three players in the same game.*
Norm 3: *Players cannot leave the game field during the game.*
Norm 4: *The referee must send off a player after (s)he has done a second caution in the same match.* In this simplified version of the soccer game, there is only one situation that characterizes a caution; a player leaving the game field before the referee has stopped it. At the first caution, the agent receives a yellow card.
Norm 5: *Kickers cannot handle the ball.*
Norm 6: *The referee must declare a penalty if kicker handle the ball.*

3 Dialogical and Non-Dialogical Actions

Non-dialogical actions are the ones not related to interactions between agents. Not all actions executed by agents in MAS provide support for sending and receiving messages between them [13]. There are actions that modify the environment (for

[1] Jess is a rule-based system. http://www.jessrules.com/

example, updating the state of a resource) that do not characterize a message being sent to or received from another agent. In the soccer game example, the actions of kicking the ball or handling it are non-dialogical actions. In addition, actions that modify the position of an agent in an environment do not characterize a dialogical action either. The actions of entering or leaving the game field are not dialogical ones.

Some actions can be defined as a dialogical or a non-dialogical one, depending on how the problem is modeled. In the soccer game, to start a game and to stop it was considered dialogical actions. Agents receive a message informing about the state of the game. The dialogical actions of the soccer game example are: to start the game, stop it, punish player, declare penalty and show the yellow card. The non-dialogical ones are: enter in the game field, leave it, handle the ball, kick the ball, substitute a player and check the player's equipment.

4 Describing Norms

Since our intention is to contribute to the work presented in [7], we extend the BNF normative language to represent non-dialogical actions and to describe conditions and time situations that are defined by those non-dialogical actions. In addition, the specification of dialogical actions already presented in the previous normative language was extended in order to be possible to describe messages attributes stated in the FIPA ACL language[2].

4.1 Specifying Non-Dialogical Actions

The original BNF description of the normative language defines norms as the composition of a *deontic* concept (characterizing obligation, prohibition or permission) and an action followed by a temporal situation and a *if* condition, when pertinent. In such definition, actions are limited to utterance of illocutions.

In our proposed extension, the *action* concept was generalized to also describe non-dialogical ones. Dialogical and non-dialogical actions are complementary, as illustrated by the grammar that specifies that these are the only two possible actions' kinds. Non-dialogical actions state the entities whose behavior is being restricted and the actions that are being regulated. Due to the way the *entity* concept was defined, a non-dialogical norm, i.e., norm that regulate non-dialogical actions, can be applied to all agents in the system, to a group of agents, to agents playing a given role or even to a unique agent.

```
<norm> ::= <deontic_concept> '(' <action> ')'
 | <deontic_concept> '(' <action><temporal_situation> ')'
 | <deontic_concept> '(' <action> IF <if_condition> ')'
 | <deontic_concept> '(' <action> <temporal_situation> IF <if_condition> ')'
<deontic_concept> ::= OBLIGED | FORBIDDEN | PEMITTED
<action>::= <non_dialogical_action> | <dialogical_action>
<non_dialogical_action> ::= <entity> 'EXECUTE' <exec>
<entity>::= <agent>':'<role> | <role> | <agent> | <group> | 'ALL'
```

[2] http://www.fipa.org/repository/aclspecs.html

In this paper we are limiting non-dialogical actions to the execution of an object/class method or to the execution of the action of an agent plan [13]. Non-dialogical norms that regulate the access to resources specify the entities that have restricted access to execute the methods of the resource. Non-dialogical norms that regulate (non-dialogical) actions not related to the access to resources describe entities that have restricted access to the execution of an action of a plan.

```
<exec> ::= <objectORclass>'.'<method>'('<parameters>')''('<contract>')'
 | <plan>':'<action>'('<parameters>')''('<contract>')'
 |...!the parameters and the contract can be omitted
```

In [13], the authors affirm that non-dialogical actions can be described as abstract actions that are not in the set of actions defined by the agents or in the set of methods of the classes. Agents must translate the actions and methods to be executed into more abstract ones. With the aim to help agents in such transformation, we propose the use of contracts. A contract is used to formally describe the behavior of the actions/methods while specifying its invariants, pre and post-conditions [11]. We do not impose any language to be used to describe the terms of a contracts[3].

```
<contract> ::= <pre>';'<post>';'<inv> |... !pre, post and inv can be omitted
<pre> ::= <expression> | <expression> <opl> <pre> ...
<opl> ::='AND' | 'OR' | 'XOR' | 'NOR'|... !pre, post and inv are similarly defined
```

Such extensions make possible to describe, for instance, norms that regulate the execution of an action while describing the parameters required for its execution and the contract that defines it. The extensions enable, for example, the definition of *norm 2*. Such norm states that a coach cannot substitute more than three players in the same game. The coach cannot execute an action that substitutes players if the number of substitutions is already 3.

```
FORBIDDEN ( coach EXECUTE managingTeam:SubstitutePlayer (outPlayer,inPlayer,team)
           ( team.coach = coach; team.substitutions = team.substitutions@pre+1 AND
             team.playersInField->excludes(outPlayer)AND
             team.playersInField->includes(inPlayer); )
   IF team.substitutions >= 3 )
```

The action governed by *norm 1* is also a non-dialogical action and states that the referee must check the players' equipment before starting the game. The action of checking the equipment is a non-dialogical action since the referee needs not to interact with the player but with its equipment. On the other hand, the action of starting a game is a dialogical action modeled as a message from the referee to everybody in the game (as will be presented in Section 4.4).

```
OBLIGED ( referee EXECUTE managingGame:checkEquipment (players)
   BEFORE ( UTTER(game; s1; INFORM(;referee;;[;gameStart;;;;;;])))  )
```

4.2 Extending the Temporal Situations

The *temporal situation* concept specified in the normative language is used to describe the period of valid (or active) norms. Norms can be activated or deactivated

[3] In this paper we are using OCL (http://www.omg.org/technology/documents/formal/ocl.htm)

due to the execution of an (dialogical or non-dialogical) action, to the change in the state of an object or an agent, to the occurrence of a deadline, and to the combination of such possibilities. In the previous normative languages the authors only consider the execution of dialogical actions and the occurrence of a deadline as temporal situations. The normative language was extended to contemplate the activation and deactivation of norms due to the execution of non-dialogical actions, to the change in the state of an object or an agent (without specifying the action that was responsible for that) and to the combination of the above mentioned factors (as specified in the *situation* concept).

```
<temporal_situation> ::= BEFORE <situation> | AFTER <situation>
| BETWEEN '(' <situation> ',' <situation> ')'
```

The extensions enable, for example, the definition of *norm 3* that states that players cannot leave the game between its initial and its interruption, as shown below.

```
FORBIDDEN ( player EXECUTE moving:LeaveField ()
                          ( agent.position@pre=inField; agent.position<>inField; )
 BETWEEN ( UTTER(game; si; INFORM(;referee;;[;gameStart;;;;;;])),
           UTTER(game; si; INFORM(;referee;;[;gameStopped;;;;;;])) ))
```

Another norm that makes use of temporal situation is *norm 4*. It states that the referee must send off a player after (s)he receives a second caution in the same match. If player leaves the field of play and (s)he has already been shown a yellow card, the referee must send him(her) off. Note that such *norm 4* is conditioned to the execution of an action governed by *norm 3* and, thus, the *after* condition is exactly *norm 3*.

```
OBLIGED ( UTTER(game;si;CAUTION(;referee;;kicker[;sentOff;;soccerGame;;;;))
  AFTER ( player EXECUTE moving:LeaveField()
                          ( agent.position@pre=inField;agent.position<>inField; )
          BETWEEN ( UTTER(game; si; INFORM(;referee;;[;gameStart;;;;;;])),
                    UTTER(game; si; INFORM(;referee;;[;gameStopped;;;;;;])) ))
  IF player.yellowCard = true )
```

4.3 Extending the IF Condition

The *if condition* defined in the original normative language is used to introduce conditions over variables, agents' observable attributes or executed dialogical actions. Therefore, by using such language it is not possible to describe *nom 6* since it is conditioned to the execution of a non-dialogical action. Our proposed extension makes possible to specify a condition related to an executed non-dialogical action or to a fired norm.

```
<if_condition> ::= <cond_expression> | NOT '(' <cond_expression> ')'
<cond_expression> ::= <condition> | NOT <condition>
 | <condition> ',' <if_condition> | NOT <condition> ',' <if_condition>
<condition> ::= <action> | <deontic_concept> '(' <action> ')' |...
```

Norm 6 defines that the referee must declare a penalty if a kicker handles the ball. The non-dialogical action of handling the ball is the *if condition* of *norm 6* and can be described as follows.

```
OBLIGED (UTTER(game; si; PENALTY(;referee;kickerTeam;[;penalty;;soccerGame;;;;]))
    IF kicker EXECUTE play:handleBall)
```

4.4 Extending Dialogical Actions

In [8], the authors represent the execution of dialogical actions by the identification of the action (not carried out yet) of submitting an illocution. In their point of view, an illocution is an information that carries a message to be sent by an agent playing a role to another agent playing another role. The *illocution* concept was extended to be possible to omit the agents that send and receive the messages. Not always will be possible to specify the agents that will send and receive the messages while describing the norms. Sometimes only the roles that those agents will be playing can be identified. Moreover, the roles of the sender and receiver can also be omitted. It may be the case that no mater the one is sending a message or no mater the one is receiving it, the norm must be obeyed.

```
<dialogical_action> ::= 'UTTER(' <scene> ';' <state> ';' <illocution> ')'
| 'UTTERED(' <scene> ';' <state> ';' <illocution> ')'
<illocution> ::= <perf>'('<sender>';'<role>';'<receiver> ';'<role>'['<msg>'])'
|...!it is possible to omit the senders, receivers and also their roles
```

Since a message can be sent to several agents, the *receiver* concept was also extended to make possible to describe the group of agents that will be the receivers of the message.

```
<sender> ::= <agent>
<receiver> ::= <agent> | <group>
```

By using the extensions provided above for illocution, it is possible to model *norms 1* (Section 4.1), *4* (Section 4.2) and *6* (Section 4.3) that omit the agent identification that is playing the referee role. In such cases, it is not important to identify the agent but only the role that the agent is playing. *Norm 1* also omits the receiver and its role to characterize that the message is being broadcasted. *Norm 4* identifies the role of the receiver but does not identify the agent playing the role since the message to be send does not depend on the agent. Moreover, *norm 6* does not identify the receiver agent but the receiver *team* that will be punished.

4.5 Specifying Messages

The *message* concept has not been specified in the previous version of the normative language. We propose to specify such concept since it may be necessary to provide some characteristics of the messages while describing the norms. The *message* concept was extended according to the parameters defined by an ACL message. While describing *norms 4* and *6* we have used the extended *message* concept to point out the ontology being used to support the interpretation of the content expression.

```
<msg> ::= <conversation_id>';'<contents>';'<language_encoding>';
'<ontology_protocol>';'<reply_by>';'<reply_to>';'<reply_with>';'<in_reply_to>
 |...!it is possible to omit any parameter.
```

5 Implementing Norms

Once we have seen how norms can be described, we need to demonstrate how they are implemented. Similar to the approach presented in [8], we have also used Jess to implement the governance mechanism. Jess is a rule-based system that maintains a collection of facts in its knowledge base. Jess was chosen due two main reasons: (i) it provides interfaces to programs in Java and (ii) it is possible to dynamically change the set of rules defined in Jess from the execution of Java programs. MAS implemented in Java can make use of the knowledge base and the declarative rules provided by Jess. Such MAS can also update the set of rules during the execution.

The use of Jess makes possible to describe facts and rules that are fired according to the stated facts. In our approach, facts are agents' observable attributes, (dialogical and non-dialogical) actions executed by the agents, the norms activated by the rules, and the information about norm violations. The rules are fired according to the executed actions or observable attributes and can activate norms or assert violations.

5.1 The Use of Jess

In Jess, facts are described based on templates that specify the structure of the facts. We have defined a template to define agents' observable attributes and three templates to describe actions: one for describe dialogical actions and two for describing the two different kinds of non-dialogical actions contemplated in the paper (method calling and execution of the action of an agent plan). Besides, we have also described nine templates for describing each of the three norm kinds (obliged, permitted and forbidden) associated with the three different actions (message, method calling and plan execution). In addition, one template was defined for being used to describe norm violations. Such template points out the norm that was violated and the facts that have violated the norm. The two examples below illustrate templates to describe an obligation norm to execute the action of a plan and a violation.

```
(deftemplate OBLIGED-non-dialogical-action-plan
    (slot entity)(slot role)(slot plan) (slot action) (slot attribs (type String))
    (slot contract-pre (type String)) (slot contract-post (type String))
    (slot contract-inv (type String)) (slot beliefUpdated (type String))
    (slot condition (type String)))

(deftemplate VIOLATION (slot norm-violated) (multislot action-done))
```

Rules are composed of two parts. The left-hand side of the rule describes patterns of facts that need to be inserted in the knowledge base in order to fire the rule. The right-hand side defines facts that will be upload to the knowledge based if the rule is fired. In our approach, these facts will be norms or norms' violations. Examples of rules are presented in Sections 5.3, 5.4, 5.5 and 5.6.

5.2 Some Guidelines

For each application norm, there is (usually) a need for describing three rules in Jess. The first rule is used to state the norm by conditioning it to the facts that activate the

norm. If the facts are inserted into the knowledge base, the rule is fired and the norm is activated. The second rule deactivates the norm retracting it from the knowledge base. The period during while some norms are active are limited and conditioned to the addition of some facts in the knowledge base. The third and final rule points out the violations. Prohibitions are violated if facts are inserted into the knowledge base during while they are forbidden and permissions are violated if the facts are inserted into the knowledge outside the period during while they are permitted. The violations of obligations occur if facts are not inserted into the knowledge base in the corresponding period. The following Sections will demonstrate how to implement those rules according to the *temporal situations* and *if conditions* mentioned in Section 4.

5.3 Simple Obligations, Permissions and Prohibitions

Norms that describe obligations, permissions or prohibitions over the execution of actions without defining any temporal situation or if condition are always active. Such norms are never deactivated no matter what happens.

Although it is possible to describe obligations and permissions over the execution of a norm without stating any condition, it is not possible to state violations. For each obligation or permission that is not associated with any temporal situation or if condition, only one rule that states the norm must be described. The obligations characterize that the actions must be executed but do not state when the executions must be checked. Permissions characterize that such actions can always be executed, and, therefore, such norms are never violated by the permitted agents. When permissions are applied to sub-sets of agents, we assume that prohibitions are stated to the ones not permitted to execute the actions. Prohibitions can do be checked and violations can be fired in case the actions are executed. Therefore, for each norm that describes prohibition for the execution of an action, two rules need to be defined: (i) to assert the prohibition; and (ii) to assert the violations if the forbidden facts are added to the knowledge base.

In order to exemplify the use of Jess we describe the implementation of *norm 5*. Rule (i) asserts the prohibition that is not conditioned to any fact. Rule (ii) asserts the violation if a kicker handles the ball.

```
(defrule forbidden:KickerHandleBall       ;(rule i)
=> (assert (FORBIDDEN-non-dialogical-action-plan (entity kicker)(plan play)
                                                 (action handleBall))))

(defrule violation:KickerHandleBall       ;(rule ii)
?fact <-(non-dialogical-action-plan (entity kicker)(plan play)(action handleBall))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (entity kicker)(plan play)
                                                    (action handleBall))
=> (assert (VIOLATION (norm-violated (fact-id ?forbidden))
                      (action-done (fact-id ?fact)))))
```

5.4 Norms Regulating Actions Executed Before the Occurrence of a Fact

Obligations for executing an action X before the occurrence of a fact W are verified testing if X has been executed before W occurs. For governing such norms three rules

are defined: rule (i) asserts the obligation for execute X; rule (ii) retracts the obligation if X has been executed and W occurs; and rule (iii) asserts a violation if W occurs but X has not been executed (what can be verified by the existence of the obligation).

Permissions for executing an action X before the occurrence of W are verified testing if X is executed after W. In such case, the execution of X is not permitted. These norms are governed by three rules: rule (i) asserts the permission for execute X; rule (ii) retracts the permission if W occurs; and rule (iii) asserts a violation if W occurs and X is executed.

Prohibitions for executing an action X before the occurrence of an action W are verified testing if X is executed and W has not occurred. Such norms are also governed by three rules: rule (i) asserts the prohibition; rule (ii) retracts the prohibition if W occurs; and rule (iii) asserts a violation if X is executed and W has not occurred (what can be verified by the existence of the prohibition). We assume that W can occur many times but obligations should be fulfilled before the first time it occurs and permissions and prohibitions are only active before its first occurrence.

Norm 1 is a good example for illustrate the implementation of norms that govern the actions that must be executed before another one. Since the norm defines that a referee is *obliged* to check the equipment of the players *before* starting the game, three rules was defined to govern such norm. Rule (i) states the obligation. Rule (ii) retracts the obligation if the referee has checked the player equipment when the game starts. Rule (iii) asserts a violation if the game has been started and the obligation still holds informing that the referee has not checked the equipment. The obligation governs a non-dialogical action that must be executed after a dialogical action.

```
(defrule obliged:CheckEquipment           ;(rule i)
 =>(assert (OBLIGED-non-dialogical-action-plan (entity referee)(plan managingGame)
     (action checkEquipment)(attribs players)
     (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))))

(defrule retract:CheckEquipment           ;(rule ii)
(non-dialogical-action-plan (entity referee)(plan managingGame)
                            (action checkEquipment)(attribs players))
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
?obliged <- (OBLIGED-non-dialogical-action-plan (ntity referee)
       (plan managingGame)(action checkEquipment)(attribs players)
       (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))
=> (retract ?obliged))

(defrule violation:CheckEquipment         ;(rule iii)
?fact <- (dialogical-action (scene game)(state si)(performative inform)
                            (sRole referee)(message "gameStart"))
?obliged <- (OBLIGED-non-dialogical-action-plan (ntity referee)
       (plan managingGame)(action checkEquipment)(attribs players)
       (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))
=> (assert (VIOLATION (norm-violated (fact-id ?obliged))
                   (action-done (fact-id ?fact)))))
```

5.5 Norms Regulating Actions Executed After the Occurrence of a Fact

Obligations for executing an action X after the occurrence of Y (or if Y occurs) cannot be governed since it is not possible to affirm that the execution of X will never occur after the execution of Y. It is not possible to state a rule that fires a violation for

such norm since the action X can be executed anytime after Y has occurred. In order to govern such norms it is necessary to state any temporal situation limiting the time for the execution of X after Y has occurred. The temporal concept *between* should be used instead of *after* or *if* for governing such obligations. *Norms 4* and *6* are example of norms that should be implemented by using *between*, as depicted in Section 5.6.

Permissions for executing X after the occurrence of Y can be governed by two rules: rule (i) assert the permission if Y occurs; and rule (ii) asserts a violation if X is executed but Y has not occurred yet (i.e., there is no permission for execute X).

The governance of prohibitions for executing X after the occurrence of Y is the opposite to the governance of the related permission. Such governance is also characterized by two rules: rule (i) asserts the prohibition if Y occurs; and rule (ii) asserts a violation if X is executed after Y has occurred or if Y is true.

In order to exemplify a norm that use the *if condition* we refer to *norm 2*. This norm defines that the coach cannot execute an action that substitutes players if the number of substitutions is equal or greater than 3. The prohibition governs a non-dialogical action that is condition to the state of an object.

```
(defrule forbidden:PlayerSubstitution     ;(rule i)
(attribute-value (objectORagent team)(attribute substitutions)(value 3))
=> (assert (FORBIDDEN-non-dialogical-action-plan (role coach)(plan managingTeam)
               (action substitutePlayer)(attribs outPlayer,inPlayer,team)
               (contract-pre "team.coach=coach")
               (contract-post "team.substitutions=team.substitutions@pre+1 AND
                               team.playersInField->excludes(outPlayer) AND
                               team.playersInField->includes(inPlayer)") )))

(defrule violation:PlayerSubstitution     ;(rule ii)
?fact1 <- (non-dialogical-action-plan (role coach)(plan managingTeam)
                                      (action substitutePlayer))
?fact2 <- (attribute-value (objectORagent team)(attribute substitutions))
?forbidden <-(FORBIDDEN-non-dialogical-action-plan (role coach)(plan managingTeam)
                (action substitutePlayer)(attribs outPlayer,inPlayer,team)
                (contract-pre "team.coach=coach")
                (contract-post "team.substitutions = team.substitutions@pre+1 AND
                                team.playersInField->excludes(outPlayer) AND
                                team.playersInField->includes(inPlayer)"))
=> (if (>= (fact-slot-value ?fact 2) 3 ) then
       (assert (VIOLATION (action-done ?fact1  ?fact2)
                          (norm-violated ?forbidden))) ))
```

5.6 Norms Regulating Actions Executed Between the Occurrence of Two Facts

A norm that states an obligation for executing an action X after the occurrence of Y and before the execution of W is governed by three rules: rule (i) asserts the obligation for execute X if Y occurs; rule (ii) retracts the obligation if X is executed and if W occurs; and rule (iii) asserts a violation if W occurs but X has not been executed.

The permission for executing X between the occurrence of Y and W is governed by the following four rules: rule (i) asserts the permission for execute X if Y occurs; rule (ii) retracts the permission if W occurs; rule (iii) asserts a violation if W occurs and X is executed; and rule (iv) asserts a violation if X is executed but Y has not occurred yet (i.e., if the permission for executing X has not been fired yet).

Prohibitions for executing X between the occurrence of Y and W are governed by three rules: rule (i) asserts the prohibition if Y occurs; rule (ii) retracts the prohibition if W occurs; and rule (iii) asserts a violation if X is executed, Y has occurred but W has not occurred, i.e., X is executed and the prohibitions is still activated. Note that the rules that govern both prohibitions and permissions while using the temporal concept *between* are the combination of the rules used to govern such norms using the *after* and *before* temporal concepts.

The use of *between* can be exemplified by *norm 3*. It states that the player is forbidden to leave the field between the beginning and the end of the game. The norm defines a prohibition to execute a non-dialogical action limited by the execution of two dialogical actions. Rule (i) asserts the prohibition if the first dialogical action is executed, rule (ii) retracts the prohibition if the second dialogical action is executed and rule (iii) declares a violation if the non-dialogical action is executed during while it is being prohibited.

```
(defrule forbidden:LeaveField     ;(rule i)
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
 => (assert (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
           (action leaveField)(contract-pre agent.position@pre=inField)
           (contract-post agent.position!=inField ))))

(defrule retract:LeaveField       ;(rule ii)
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStop"))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
                (action leaveField)(contract-pre agent.position@pre=inField)
                (contract-post agent.position!=inField ))
 => (retract ?forbidden))

(defrule violation:LeaveField     ;(rule iii)
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
                (action leaveField)(contract-pre agent.position@pre=inField)
                (contract-post agent.position!=inField ))
?fact <- (non-dialogical-action-plan (role player)(plan moving)(action leaveField)
           (contract-pre agent.position@pre=inField)
           (contract-post agent.position!=inField ))
=> (assert (VIOLATION (norm-violated (fact-id ?forbidden))
                   (action-done (fact-id ?fact)))))
```

Sections 5.3 and 5.5 point out that some obligations over the execution of a norm that cannot be governed. Since obligations need not to be fulfilled immediately after they were declared, it is necessary to inform the period during while the agents are being obligated to execute the action in order to govern them. *Norms 6* and *4* are very good examples of such obligations. *Norm 6*, for instance, defines that the referee must declare a penalty if a kicker handles the ball. However, this norm does not define how much time does the referee has to fulfill its obligation. Therefore, it is not possible to affirm that the obligation was not fulfilled since it can be at any time. In order to properly regulate such norm it is needed to provide a limit till when this obligation must be fulfilled. *Norms 6* was adapted to inform that the referee has 1 minute to declare the penalty after the kicker has handled the ball.

```
OBLIGED ( UTTER(game; si; PENALTY(;referee;kickerTeam;[;penalty;;soccerGame;;;;]))
   BETWEEN ( kicker EXECUTE play:handleBall, 1 MINUTES OF kicker EXECUTE
             play:handleBall ) )
```

6 Conclusion

This paper proposes the implementation of norms[4] that govern dialogical and non-dialogical actions by using Jess. The governance system proposed in [6] receives (not always true) testimonies about executed actions that are related to norm violations. After judging the testimonies and concluding that the actions really were executed, such information is uploaded to the Jess knowledge-based. The set of Jess rules are, then, checked and the related norms and violations are fired. The fired norm or violation is also facts accumulated in the Jess database. We have implemented in Jess at least one norm taking into account the three *deontic* concepts, the proposed temporal situations and if conditions presented in the paper by using the soccer game.

Although the current version does not contemplate sanctions and awards, it can be easily extended in order to do so. The sanctions should be provided when the related violations are fired. The awards should be supplied when the norms are retracted and no violation of such norms has been fired. In addition, a (semi)automatic approach for generating Jess rules according to the norms specified by the use of the normative language could be developed.

References

1. Boella, G.; van der Torre, L.: Regulative and Constitutive Norms in Normative Multi-Agent Systems. In Proceeding of KS. AAAI Press, (2004) 255-265.
2. Artikis, A., Kamara, L., Pitt, J., Sergot, M.: A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. LNCS 3476. Springer-Verlag (2005) 221-238.
3. Broersen, J., Dignum, F., Dignum, V. and Meyer, J. Designing a deontic logic of deadlines. In 7th Int. Workshop of Deontic Logic in Computer Science. Springer-Verlag(2004) 43-56.
4. Cranefield, S.: A Rule Language for Modelling and Monitoring Social Expectations in Multi-Agent Systems. LNCS 3913. Springer-Verlag, (2006) 246-258.
5. Dignum, F., Broersen, J., Dignum, V., and Meyer, J. Meeting the deadline: Why, when and how. LNCS 3228, Springer-Verlag (2004) 30-40
6. Duran, F.; Silva, V.; Lucena, C. "Using Testimonies to Enforce the Behavior of Agents" In Proceedings of Workshop COIN at AAMAS (2007) (in current proceedings).
7. García-Camino,A. Rodríguez-A,J, Sierra,C, Vasconcelos,W. Norm-Oriented Programming of Electronic Institutions. In Proceedings of AAMAS,ACM Press(2006)670-672
8. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: Implementing Norms in Electronic Institutions. In: Proceedings of AAMAS, ACM Press (2005) 667-673.
9. López, F.: Social Power and Norms: Impact on agent behavior. PhD thesis, Univ. of Southampton (2003)
10. López, F, Luck, M. and d'Inverno,M. Constraining autonomy through norms. In Proceedings of AAMAS, ACM Press (2002) 674-681
11. Meyer, B. Object-Oriented Software Construction Prentice Hall, second edition (1997)
12. Singh, M.: An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law v. 7 (1), Springer (1999) 97-113.
13. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Implementing Norms in Multiagent Systems. LNAI 3187. Springer-Verlag (2004) 313 - 327

[4] The full normative language described in the paper and the Jess program used to illustrate our approach are available at http://maude.sip.ucm.es/~viviane/products.html.

Ignoring, Forcing and Expecting Concurrent Events in Electronic Institutions

Andrés García-Camino

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB, 08193 Bellaterra, Spain
andres@iiia.csic.es

Abstract. Norms constitute a powerful coordination mechanism among heterogeneous agents. We propose means to specify open environments regulated using the notions of ignoring, forcing, expecting and sanctioning events and prevention of unwanted states. These notions make explicit and clear the stance of institutions about forbidden and obligatory behaviour. Our rule-based language calculates the effects of concurrent events generated by agents given a set of norms based on the deontic notions previously mentioned. Our formalism has been conceived as basis for an implementation of Electronic Institutions.

1 Introduction

Ideally, open multi-agent systems (MAS) involve heterogeneous and autonomous agents whose concurrent interactions ought to conform to some shared conventions. The challenge is how to express and enforce such conditions so that truly autonomous agents can adscribe to them. One way of addressing this issue is to look at MAS as environments regulated by some sort of normative framework.

There are many examples of languages for regulating agent behaviour (for example, [1–5]). However, very few of them regulate concurrent events taking into account the rest of events that occur at an instant of time. The few that exist (e.g. [3]) are not conceived to deal with open MAS.

Furthermore, in the literature we find that almost all these languages are based on deontic logic [6] that establishes which actions are permitted, forbidden or obligatory. However, it does not establish which is the semantics of these modalities with respect to a computational system. For instance, when an action is claimed to be forbidden, does it means that it is prevented to happen, or that the agents that bring it about must be sanctioned or that the effects of that action are just ignored?

Instead, we propose a language, called $\mathcal{I}$, and one implementation of it that uses the notions of ignoring, forcing, and expecting events along with the notion of preventing a state, in the computation of the effects of concurrent agent behaviour in a regulated open MAS. The main contributions of $\mathcal{I}$ is the management of sets of events that occur simultaneously and the distinction between

norms that can be violated or not. For instance, an obligation that may be violated to perform a set of simultaneous events is represented as the expectation of the attempts to perform them. However, the enforcement of an obligation that may not be violated to perform a set of events is carried out by the system by taking these events as performed even they are not. We denote such enforcement as forcing events.

The paper is structured as follows. Section 2 introduces $\mathcal{I}$, a rule language for electronic institutions. A basic example illustrating the expressiveness of $\mathcal{I}$ is shown in section 3. In section 4, we introduce the formulae that we use for modelling electronic institutions. An example of a bank institution is presented in section 5. In section 6 we contrast our approach with a sample of other contemporary work. Finally, we draw conclusions and outline future work in section 7.

2 $\mathcal{I}$: A Rule Language for Electronic Institutions

In this section we introduce a rule language for the regulation and management of concurrent events generated by a population of agents. Our rule-based language allows us to represent norms and changes in an elegant way.

The building blocks of our language are first-order terms and implicitly, universally quantified atomic formulae without free variables. We shall make use of numbers and arithmetic functions to build terms; arithmetic functions may appear infix, following their usual conventions[1]. We also employ arithmetic relations (*e.g.*, $=$, $\neq$, and so on) as predicate symbols, and these will appear in their usual infix notation with their usual meaning.

$$
\begin{array}{rl}
\textit{ECA-Rule} & ::= \textbf{on}\ \textit{events}\ \textbf{if}\ \textit{conditions}\ \textbf{do}\ \textit{actions} \\
\textit{if-Rule} & ::= \textbf{if}\ \textit{conditions}\ \textbf{do}\ \textit{actions} \\
\textit{ignore-Rule} & ::= \textbf{ignore}\ \textit{events}\ \textbf{if}\ \textit{conditions} \\
\textit{prevent-Rule} & ::= \textbf{prevent}\ \textit{conditions}\ \textbf{if}\ \textit{conditions} \\
\textit{force-Rule} & ::= \textbf{force}\ \textit{events}\ \textbf{on}\ \textit{events}\ \textbf{if}\ \textit{conditions}\ \textbf{do}\ \textit{actions} \\
\textit{events} & ::= \textit{list\_of\_events} \mid \emptyset \\
\textit{list\_of\_events} & ::= \textit{atomic\_formula}, \textit{list\_of\_events} \mid \textit{atomic\_formula} \\
\textit{conditions} & ::= \textit{conditions} \wedge \textit{conditions} \mid \neg(\textit{conditions}) \mid \textit{atomic\_formula} \\
\textit{actions} & ::= \textit{action} \bullet \textit{actions} \mid \textit{action} \\
\textit{action} & ::= \oplus \textit{atomic\_formula} \mid \ominus \textit{atomic\_formula}
\end{array}
$$

Fig. 1. Grammar for $\mathcal{I}$

One goal of the $\mathcal{I}$ language is to specify which are the effects of concurrent events and this is achieved with Event-Condition-Action (ECA) rules. Intuitively, an ECA-rule means that whenever the events occur and the conditions hold then the actions are applied. These actions consist in the addition and removal of atomic formulae from the state of affairs. ECA-rules are checked in parallel and they are executed only once without chaining.

[1] We adopt Prolog's convention using strings starting with a capital letter to represent variables and strings starting with a small letter to represent constants.

If-rules are similar to rules in standard production systems, if the conditions hold then the actions are applied. They are implemented with a forward chaining mechanism: they are executed sequentially until no new formula is added or removed.

Ignore-rules are used for ignoring events when the conditions hold in order to avoid unwanted behaviour. Similarly, prevent-rules are used for preventing some conditions to hold in the situations given. In order to prevent unwanted states, events causing such unwanted states are ignored. Force-rules generate events and execute actions as consequence of other events and conditions.

Sanctions over unwanted events can be carried out with ECA-rules. For instance, we can decrease the credit of one agent by 10 if she generates certain event.

We add an additional kind of rules, expectation-rules, that generate and remove expectations of events. If the expectation fails to be fulfilled then some sanctioning or corrective actions are performed.

$$\begin{aligned} \textit{expectation-Rule} ::= \ & \textbf{expected } \textit{event} \textbf{ on } \textit{events} \textbf{ if } \textit{conditions} \\ & \textbf{fulfilled-if } \textit{conditions}' \textbf{ violated-if } \textit{conditions}'' \\ & \textbf{sanction-do } \textit{actions} \end{aligned}$$

Each expectation rule can be translated into three ECA-rules:

$$\textbf{on } \textit{events} \textbf{ if } \textit{conditions} \textbf{ do } \oplus \textit{exp}(\textit{event}) \tag{1}$$

$$\textbf{if } \textit{exp}(\textit{event}) \wedge \textit{conditions}' \textbf{ do } \ominus \textit{exp}(\textit{event}) \tag{2}$$

$$\textbf{if } \textit{exp}(\textit{event}) \wedge \textit{conditions}'' \textbf{ do } \ominus \textit{exp}(\textit{event}) \bullet \textit{actions} \tag{3}$$

Rules 1 and 2 respectively adds and removes an expectation whenever the events have occurred and the conditions hold. Rule 3 cancels the unfulfilled expectation and sanctions an agent for the unfulfilled expectation by executing the given *actions* whenever some *conditions* hold.

2.1 Semantics

Instead of basing the $\mathcal{I}$ language in the standard deontic notions, two types of prohibitions and two types of obligations are included. In our language, ECA-rules determine what is possible to perform, i.e. they establish the effects (including sanctions) in the institution after performing certain (possibly concurrent) events. ECA-rules might be seen as conditional count-as rules: the given events count as the execution of the actions in the ECA-rule if the conditions hold and the event is not explicitly prohibited. As for the notion of permission, all the events are permitted if not explicitly prohibited. The notion of an event being prohibited may be expressed depending on whether that event has to be ignored or not. If not otherwise expressed, events are not ignored. Likewise, the notion of a state being prohibited may be specified depending on whether that state has to be prevented or not. By default, states are not prevented. Obligations are differentiated in two types: expectations, which an agent may not fulfill, and forced (or obligatory) events, which the system takes as institutional events even they are not really performed by the agents.

Each set of ECA-rules generates a labelled transition system $\langle S, E, R\rangle$ where each state S is a set of atomic formulae, E is a set of events, and R is a $S \times 2^E \times S$ relationship indicating that whenever a set of events occur in the former state, then there is a transition to the subsequent state.

Ignore-rules avoid to execute any transition that contains in its labelling all the events that appear in each ignore-rule. For instance, having a rule **ignore** α_1 **if** *true* would avoid to execute the transitions labelled as $\{\alpha_1\}$, $\{\alpha_1, \alpha_2\}$ and $\{\alpha_1, \alpha_2, \alpha_3\}$. However, having a rule **ignore** α_1, α_2 **if** *true* would avoid to execute $\{\alpha_1, \alpha_2\}$ and $\{\alpha_1, \alpha_2, \alpha_3\}$ but not $\{\alpha_1\}$.

Prevent-rules ignore all the actions in an ECA-rule if it brings the given formulae about. For example, suppose that we have

$$\textbf{prevent } q_1 \textbf{ if } \textit{true}$$

along with ECA-rules 4, 5 and 6. After the occurrence of events α_1 and α_2 and since q_1 is an effect of event α_2, all the actions in ECA-rule 5 would be ignored obtaining a new state where p and r hold but neither q_1 nor q_2.

$$\textbf{on } \alpha_1 \textbf{ if } \textit{true} \textbf{ do } \oplus p \tag{4}$$

$$\textbf{on } \alpha_2 \textbf{ if } \textit{true} \textbf{ do } \oplus q_1 \bullet \oplus q_2 \tag{5}$$

$$\textbf{on } \alpha_1, \alpha_2 \textbf{ if } \textit{true} \textbf{ do } \oplus r \tag{6}$$

Force-rules generate events during the execution of the transition system. However, the effects of such events are still specified by ECA-rules and subject to prevent and ignore-rules.

2.2 Operational Semantics

In the definitions below we rely on the concept of *substitution*, that is, the set of values for variables in a computation [7, 8]:

We now define the semantics of the conditions, that is, when a condition holds:

Definition 1. *Relation* $\mathbf{s}_l(\Delta, C, \sigma)$ *holds between state* Δ*, a condition* C *in an* **if** *clause and a substitution* σ *depending on the format of the condition:*

1. $\mathbf{s}_l(\Delta, C \wedge C', \sigma)$ *holds iff* $\mathbf{s}_l(\Delta, C, \sigma')$ *and* $\mathbf{s}_l(\Delta, C' \cdot \sigma', \sigma'')$ *hold and* $\sigma = \sigma' \cup \sigma''$.
2. $\mathbf{s}_l(\Delta, \neg C, \sigma)$ *holds iff* $\mathbf{s}_l(\Delta, C, \sigma)$ *does not hold.*
3. $\mathbf{s}_l(\Delta, seteq(L, L2), \sigma)$ *holds iff* $L \subseteq L2$, $L2 \subseteq L$ *and* $\mid L \mid = \mid L2 \mid$.
4. $\mathbf{s}_l(\Delta, true, \sigma)$ *always holds.*
5. $\mathbf{s}_l(\Delta, \alpha, \sigma)$ *holds iff* $\alpha \cdot \sigma \in \Delta$.

Case 1 depicts the semantics of atomic formulae and how their individual substitutions are combined to provide the semantics for a conjunction. Case 2 introduces the negation by failure. Case 3 compares if two lists have the same elements possibly in different order. Case 4 gives semantics to the keyword "*true*". Case 5 holds when an atomic formulae α is part of the state of affairs.

We now define the semantics of the actions of a rules:

Definition 2. *Relation* $\mathbf{s}_r(\Delta, A, \Delta')$ *mapping a state* Δ*, the action section of a rule and a new state* Δ' *is defined as:*

1. $\mathbf{s}_r(\Delta, (A \bullet As), \Delta')$ *holds iff both* $\mathbf{s}_r(\Delta, A, \Delta_1)$ *and* $\mathbf{s}_r(\Delta_1, As, \Delta')$ *hold.*
2. $\mathbf{s}_r(\Delta, \oplus\alpha, \Delta')$ *holds iff*
   (a) $\alpha \notin \Delta$ *and* $\Delta' = \Delta \cup \{\alpha\}$ *or;*
   (b) $\Delta' = \Delta$.
3. $\mathbf{s}_r(\Delta, \ominus\alpha, \Delta')$ *holds iff*
   (a) $\alpha \in \Delta$ *and* $\Delta' = \Delta \setminus \{\alpha\}$ *or;*
   (b) $\Delta' = \Delta$.

Case 1 decomposes a conjunction and builds the new state by merging the partial states of each update. Case 2 and 3 cater respectively for the insertion and removal of atomic formulae α.

We now define relation $check_{prv}$ that checks if there is no prevent-rule that has been violated, i.e., not all the conditions hold in the state of affairs Δ. It checks whether Δ contain all the conditions of each prevent-rule or not.

Definition 3. *Relation* $check_{prv}(\Delta, PrvRules)$ *mapping a state* Δ *and a sequence PrvRules of prevent-rules holds iff an empty set is the largest set of conditions* C *such that prevent-rule* $p =$ **prevent** C **if** C', $p \in PrvRules$, $\mathbf{s}_l(\Delta, C)$ *and* $\mathbf{s}_l(\Delta, C')$ *hold.*

Definition 4. *Relation* $fire(\Delta, PrvRules, \textbf{if } C \textbf{ do } A, \Delta')$ *mapping a state* Δ*, a sequence PrvRules of prevent-rules, an if-rule and a new state* Δ' *holds iff* $assert(fired(C))$, $\mathbf{s}_r(\Delta, A, \Delta')$ *and* $check_{prv}(\Delta', PrvRules)$ *hold.*

Relation *can_fire* checks whether the conditions of a given if-rule hold and the rule after applying substitution σ has not been already fired.

Definition 5. *Relation* $can\_fire(\Delta, \textbf{if } C \textbf{ do } A, \sigma)$ *mapping a state* Δ *an if-rule and a substitution* σ *holds iff* $\mathbf{s}_l(\Delta, C, \sigma)$ *holds and* $fired(C \cdot \sigma)$ *does not hold.*

Relation *resolve* determines the rule that will be fired by selecting the first rule in the list.

Definition 6. *Relation* $resolve(RuleList, SelectedRule)$ *mapping a list of if-rules and a selected if-rule holds iff*

1. $RuleList = \emptyset$ *and* $SelectedRule = \emptyset$*; or*
2. $RuleList = \langle r_1, \cdots, r_n \rangle$ *and* $SelectedRule = r_1$.

Relation *select_rule* determines the rule that will be fired by selecting all the rules that can fire and resolving the conflict with relation *resolve*.

Definition 7. *Relation* $select\_rule(\Delta, IfRulesList, SelectedRule)$ *mapping a state of affairs* Δ *a list of if-rules and a selected if-rule holds iff Rs is the largest set of rules* $R \in IfRulesList$ *such that* $can\_fire(\Delta, R, \sigma)$*;* $resolve(Rs, SR)$ *hold and* $SelectedRule = SR \cdot \sigma$.

Relation $\mathbf{s}_{if}$ determines the new state of affairs after applying a set of if-rules to a initial state of affairs taking into account a set of prevent-rules.

Definition 8. *Relation* $\mathbf{s}_{if}(\Delta, IfRules, PrvRules, \Delta')$ *mapping a state of affairs* Δ*, a list of if-rules, a list of prevent-rules and a new state of affairs holds iff*

1. *select_rule*$(\Delta, IfRules, R)$ *hold,* $R \neq \emptyset$*,* *fire*$(\Delta, PrvRules, R, \Delta'')$ *and* $\mathbf{s}_{if}(\Delta'', IfRules, PrvRules, \Delta')$ *hold; or*
2. *select_rule*$(\Delta, IfRules, R)$ *hold,* $R = \emptyset$*; or*
3. $\mathbf{s}_{if}(\Delta, IfRules, PrvRules, \Delta')$ *hold.*

Relation *ignored* determines a set of events that occurred have to be ignored taking into account a list of ignore-rules.

Definition 9. *Relation* $ignored(\Delta, \Xi, E, IgnRules)$ *mapping a state of affairs* Δ*, a list* Ξ *of events that occurred, a list of events in a ECA-rule and a list of ignore-rules holds iff* $i =$ **ignore** E' **if** C*,* $i \in IgnRules$*,* $E' \subseteq \Xi$*,* E *intersects with* E' *and* $\mathbf{s}_l(\Delta, C)$ *holds.*

Relation $\mathbf{s}'_r$ applies $\mathbf{s}_r$ first and then $\mathbf{s}_{if}$ in order to activate the forward chaining.

Definition 10. *Relation* $\mathbf{s}'_r(\Delta, IfRules, PrvRules, ActionList, \Delta')$ *mapping a state of affairs* Δ*, a list of if-rules, a list of prevent-rules, a list of actions and a new state of affairs holds iff*

1. $ActionList = \emptyset$ *and* $\Delta' = \Delta$*; or*
2. $ActionList = \langle a_1, \cdots, a_n \rangle$*,* $\mathbf{s}_r(\Delta, a_1, \Delta'')$*,* $check_{prv}(\Delta'', PrvRules)$*,* $\mathbf{s}_{if}(\Delta'', IfRules, PrvRules, \Delta''')$ *and* $\mathbf{s}'_r(\Delta''', IfRules, PrvRules, \langle a_2, \cdots, a_n \rangle, \Delta')$ *hold; or*
3. $\mathbf{s}'_r(\Delta, IfRules, PrvRules, \langle a_2, \cdots, a_n \rangle, \Delta')$*.*

Relation $\mathbf{s}_{on}$ calculates the new state of affairs Δ' from an initial state Δ and a set Ξ of events that occurred applying a list of ECA-rules, if-rules, ignore-rules and prevent-rules.

Definition 11. *Relation* $\mathbf{s}_{on}(\Delta, \Xi, ECARules, IfRules, IgnRules, PrvRules, \Delta')$ *mapping a state of affairs* Δ*, a list* Ξ *of events that occurred, a list of ECA-rules, a list of if-rules, a list of ignore-rules, a list of prevent-rules, and a new state of affairs holds iff* As *is the largest set of actions* $A' = A \cdot \sigma$ *in a ECA-rule* $r =$ **on** E **if** C **do** A *such that* $R \in ECARules$*,* $E \cdot \sigma' \subseteq \Xi$*,* $\mathbf{s}_l(\Delta, C, \sigma'')$ *hold,* $ignored(\Delta, \Xi, E, IgnRules)$ *does not hold and* $\sigma = \sigma' \cup \sigma''$*; and* $\mathbf{s}'_r(\Delta, IfRules, PrvRules, As, \Delta')$ *hold.*

Relation $\mathbf{s}_f$ calculates the new state of affairs Δ' and the new set Ξ' of occurred events from an initial state Δ and a set Ξ of events that occurred applying a list of if-rules, ignore-rules, prevent-rules and force-rules.

Definition 12. *Relation* $\mathbf{s}_f(\Delta, \Xi, IfRules, IgnRules, PrvRules, FrcRules, \Xi', \Delta')$ *mapping a state of affairs* Δ*, a list* Ξ *of events that occurred, a list of if-rules, a list of ignore-rules, a list of prevent-rules, a list of force-rules, a new list of events that occured and a new state of affairs holds iff EAs is the largest set of tuples* $\langle FE \cdot \sigma, A \cdot \sigma \rangle$ *of forced events and actions in a force rule* $fr =$ **force** FE **on** E **if** C **do** A *such that* $fr \in FrcRules$*,* $E \cdot \sigma' \subseteq \Xi$*,* $\mathbf{s}_l(\Delta, C, \sigma'')$ *holds,ignored*$(\Delta, \Xi, E, IgnRules)$ *does not hold and* $\sigma = \sigma' \cup \sigma''$*; Es is the largest set of forced events Ev such that* $\langle Ev, A \rangle \in EAs$*;* $\Xi' = \Xi \cup Es$*; As is the largest set of actions A such that* $\langle Ev, A \rangle \in EAs$*; and* $\mathbf{s}'_r(\Delta, IfRules, PrvRules, As, \Delta')$ *holds.*

Relation $\mathbf{s}^*$ calculates the new state of affairs Δ' from an initial state Δ and a set Ξ of events that occurred applying a list of ECA-rules, if-rules, ignore-rules, prevent-rules and force-rules.

Definition 13. *Relation* $\mathbf{s}^*(\Delta, \Xi, ECARules, IfRules, IgnRules, PrvRules, FrcRules, \Delta')$ *mapping a state of affairs* Δ*, a list* Ξ *of events that occurred, a list of ECA-rules, a list of if-rules, a list of ignore-rules, a list of prevent-rules, a list of force-rules and a new state of affairs holds iff Cs is the largest set of conditions C such that retract*$(fired(C))$ *holds; assert*$(fired(false))$*,* $\mathbf{s}_{if}(\Delta, IfRules, PrvRules, \Delta'')$*,* $\mathbf{s}_f(\Delta'', \Xi, IfRules, IgnRules, PrvRules, FrcRules, \Xi', \Delta''')$ *and* $\mathbf{s}_{on}(\Delta''', \Xi', ECARules, IfRules, IgnRules, PrvRules, \Delta')$ *hold.*

3 Example of Concurrency: Soup Bowl Lifting

In this section we present an example on how to use the $\mathcal{I}$ language in order to specify a variation of a problem about concurrent action: the Soup Bowl Lifting problem [9]. Picture a situation where a soup bowl has to be lifted by two (physical) agents; one lifting from the right-hand side and the other one from the left-hand side. If both sides are not lifted simultaneously then the soup spills.

The order in which the rules are declared is important since they are executed in the order they are declared. We do not obtain the same effect with rules 7, 8 and 9 (finally *spilled* does not hold after lifted from both sides simultaneously) than with rules 9, 7 and 8 (finally *spilled* holds even after lifted from both sides simultaneously).

$$\textbf{on } pushLeft \textbf{ if } true \textbf{ do } \oplus spilled \tag{7}$$

$$\textbf{on } pushRight \textbf{ if } true \textbf{ do } \oplus spilled \tag{8}$$

$$\textbf{on } pushLeft, pushRight \textbf{ if } true \textbf{ do } \ominus spilled \bullet \ominus onTable \tag{9}$$

Rules 7 and 8 specify that the soup is spilled whenever the bowl is lifted either from the right-hand side or the left-hand side. However, rule 9 removes the spill effect whenever both events are done simultaneously. However, with rules 9, 7 and 8, we do not obtain the desired result since the *spilled* formula may be added after executing the rule that removes *spilled* formula.

To prevent the bowl from spilling, we may add the next rule to rules 7-9:

$$\textbf{prevent } \textit{spilled} \textbf{ if } \textit{true} \tag{10}$$

However, adding the following rules instead would also prevent the bowl to be lifted since ignoring one event will prevent all the combined events to be considered.

$$\textbf{ignore } \textit{pushLeft} \textbf{ if } \textit{true} \tag{11}$$

$$\textbf{ignore } \textit{pushRight} \textbf{ if } \textit{true} \tag{12}$$

Contrarily, if we add rule 13 to rules 7-9, we prevent the bowl to be lifted from both sides simultaneously but not to be only lifted from one side since we are only ignoring the events if they occur together.

$$\textbf{ignore } \textit{pushLeft}, \textit{pushRight} \textbf{ if } \textit{true} \tag{13}$$

This basic example give us a sample of the expressiveness of $\mathcal{I}$. In the next section, we introduce electronic institutions and the meaning of the formulae needed for representing them in $\mathcal{I}$.

4 Electronic Institutions

Our work extends *electronic institutions* (EIs) [10][2], providing them with a normative layer specified in terms of ignore, prevent and force rules. There are two major features in EIs: the *states* and *illocutions* (*i.e.*, messages) uttered (*i.e.*, sent) by those agents taking part in the EI. The states are connected via edges labelled with the illocutions that ought to be sent at that particular point in the EI. Another important feature in EIs are the agents' *roles*: these are labels that allow agents with the same role to be treated collectively thus helping engineers abstract away from individuals. We define below the class of illocutions we aim at – these are a special kind of term:

Definition 14. *Illocutions* I *are terms* $ill(p, ag, r, ag', r', \tau)$ *where* p *is a performative (*e.g. inform *or* request*);* ag, ag' *are agent identifiers;* r, r' *are role labels; and* τ *is a term with the actual content of the message.*

We shall refer to illocutions that may have uninstantiated (free) variables as *illocution schemes*, denoted by $\bar{\mathsf{I}}$.

An institutional state is a state of affairs that stores all utterances during the execution of a MAS, also keeping a record of the state of the environment, all observable attributes of agents and all the expectations associated with the agents.

We differentiate two kinds of events, with the following intuitive meanings:

1. I – an agent uttered illocution I.
2. $newtick(t)$ – a new tick of the clock occurred at time t.

[2] EI scenes are basically covered with ECA rules

We shall use event 2 above to obtain the time with which illocutions and expectations are time-stamped.

We differentiate two kinds of atomic formulae in our institutional states Δ, with the following intuitive meanings:

1. $inst(\mathsf{I}, t)$ – I was accepted as an institutional utterance at time t.
2. $exp(\overline{\mathsf{I}}, t)$ – $\overline{\mathsf{I}}$ is expected to be uttered since time t.

We allow agents to utter whatever they want (via I events). However, the unwanted utterances may be discarded and/or may cause sanctions, depending on the deontic notions we want or need to implement via our rules. The *inst* formulae are thus *confirmations* of the I events. We shall use formula 2 above to represent expectations of agents within EIs.

5 Applied Example: Bank

In this section we introduce an example of banking institution where agents are allowed to do certain operations with money. The operations in our bank are depositing, withdrawing and transferring. In our example we have two types of accounts called a and b owned by two different agents. In order to perform an operation in one of these accounts both agents have to *simultaneously* make the proper request.

Type a accounts have the limitation that no withdrawing, transferring from and debiting is allowed having a negative credit. If it is the case and there is enough money in a type b account of the same agent then necessary credit is automatically transferred to the account with negative credit and a fee is debited.

Type b accounts have the following limitations:

1. They cannot be in red. All the transactions that would finish in negative credit are rejected.
2. Withdrawing from or depositing to these accounts is not allowed.

Rule 14 specify the effects of opening an account of type T to agents $A1$ and $A2$ with an amount M of credit if another account of the same type with the same owners is not already opened.

$$\begin{array}{ll} \textbf{on} & newtick(Time), open\_account(Id, A1, A2, T, M) \\ \textbf{if} & \neg account(Id, A1, A2, T, \_) \wedge \neg account(Id, A2, A1, T, \_) \\ \textbf{do} & \oplus account(Id, A1, A2, T, M)\bullet \\ & \oplus inst(open\_account(Id, A1, A2, T, M), Time) \end{array} \tag{14}$$

Rule 15 specify the effect of withdrawing a given quantity M_q of money from a given account due to the simultaneous request of both owners of the account. The rules in the action section calculate the new credit for the account and modifies its value by removing the old credit and adding the new one. Likewise, a rule for the effects of depositing may also be specified.

$$\begin{array}{ll} \textbf{on} & newtick(Time), withdraw(A1, Id, M_q), withdraw(A2, Id, M_q) \\ \textbf{if} & account(Id, A1, A2, T, M) \\ \textbf{do} & M2 = M - M_q \bullet \ominus account(Id, A1, A2, T, M)\bullet \\ & \oplus account(Id, A1, A2, T, M2) \bullet \oplus inst(withdraw(A1, A2, Id, M_q), Time) \end{array} \tag{15}$$

Rule 16 specifies the effect of transferring from one account (of an agent and of a certain type) to another account possibly as payment of a certain concept C: the source account is reduced and the destination account is increased by the stated amount.

$$\begin{array}{ll} \textbf{on} & \mathit{newtick}(\mathit{Time}), \mathit{transfer}(A1, Id_s, Id_d, C, M), \mathit{transfer}(A2, Id_s, Id_d, C, M) \\ \textbf{if} & \mathit{account}(Id_s, A1, A2, T_s, M_s) \wedge \mathit{account}(Id_d, A3, T_d, M_d) \\ \textbf{do} & M2_s = M_s - M \bullet \ominus \mathit{account}(Id_s, A1, A2, T_s, M_s) \bullet \\ & \oplus \mathit{account}(Id_s, A1, A2, T_s, M2_s) \bullet M2_d = M_d + M \bullet \\ & \ominus \mathit{account}(Id_d, A3, T_d, M_d) \bullet \oplus \mathit{account}(Id_d, A3, T_d, M2_d) \bullet \\ & \oplus \mathit{inst}(\mathit{transfer}(A1, A2, Id_s, Id_d, C, M), \mathit{Time}) \end{array} \quad (16)$$

To avoid concurrent actions affecting the same account, we use rule 17. In this case, only the first action is taken into account and the rest of concurrent actions are ignored.

$$\textbf{prevent } \mathit{account}(I, A_1, A_2, T, M) \wedge \mathit{account}(I, A_1, A_2, T, M_2) \textbf{ if } M \neq M_2 \quad (17)$$

In our example, accounts of type a have the restriction that agents are not allowed to withdraw or transfer from a accounts with negative credit. This is achieved with rules like:

$$\textbf{ignore } \mathit{withdraw}(A, Id, \_) \textbf{ if } \mathit{account}(Id, A, \_, a, M) \wedge M < 0 \quad (18)$$

$$\textbf{ignore } \mathit{transfer}(A, Id_s, \_, \_, \_) \textbf{ if } \mathit{account}(Id_s, A, \_, a, M) \wedge M < 0 \quad (19)$$

Accounts of type b also have some restrictions. First, they cannot go into negative numbers. This is achieved with the following rule:

$$\textbf{prevent } \mathit{account}(Id, A1, A2, b, M) \textbf{ if } M < 0$$

Second, agents are not allowed to withdraw from accounts of type b. This is achieved by rule 20.

$$\textbf{ignore } \mathit{withdraw}(\_, Id, \_) \textbf{ if } \mathit{account}(Id, \_, \_, b, \_) \quad (20)$$

Furthermore, if an account of type a goes into the negatives then the necessary amount to avoid this situation is transferred from an account of type b. Rule 21 forces this type of events. Notice that a similar rule but with the order of the owners of the accounts reversed is also necessary since the owners may not appear in the same order.

$$\begin{array}{ll} \textbf{force} & \mathit{transfer}(A, Id_b, Id_a, a\_negative, C), \mathit{transfer}(A2, Id_b, Id_a, a\_negative, C) \\ \textbf{if} & \mathit{account}(Id_a, A, A2, a, C2) \wedge C2 < 0 \wedge C = C2 \wedge \\ & \mathit{account}(Id_b, A, A2, b, C3) \wedge C3 \geq C \end{array} \quad (21)$$

6 Related Work

In the model of Electronic Institutions of [10], agent interaction is brought about by uttering illocutions and it is decomposed in a set of scenes where only one illocution is accepted as legal simultaneously. As for norms, agents may be expected to utter certain illocutions under given conditions. However, there is no

notion of prevention of a state or force of events. Furthermore, only events that are not part of the protocol are ignored, not allowing to write further conditions in which an illocution is ignored.

The work presented in this paper is the result of the evolution of our previous work on norm languages for electronic institutions [1]. In that work, we presented a rule language that does not use forward chaining to calculate the effects of events and to explicitly manage normative positions (i.e. permissions, prohibitions and obligations). For the present work, we use those rules in the form of event-condition-action rules. Then, we added standard condition-action rules that use forward chaining. Furthermore, we changed our standard deontic notions that only regulated one illocution into more institutional-centred notions as ignoring, forcing or expecting concurrent events or preventing an institutional state.

$n\mathcal{C}_+$ is a language for representing and reasoning about action domains that include some normative notions [3]. Its semantics is based on labelled transition systems. The language allows to make queries about the transition system generated from an action description allowing to pre-dict, post-dict, or plan. In the normative aspect, $n\mathcal{C}_+$ only labels states and actions as green or red without including our notion of prevention that ignores actions that lead to an unwanted state. We can obtain this labeling by adding *green* to the initial state and rules of the form "**on** *events* **if** *conditions* **do** $\ominus$*green* $\bullet$ $\oplus$*red*" or "**if** *condition* **do** $\ominus$*green* $\bullet$ $\oplus$*red*". Instead of using ignore-rules, $n\mathcal{C}_+$ may label events as non-executable obtaining no solution when this kind of events occur. Since we want to maintain the state of the multi-agent system, we would need to ignore all the actions that occurred in that moment even the ones that does not lead to an unwanted state.

The implementation of $n\mathcal{C}_+$ loads the full transition system in order to resolve the queries. When dealing with fluents with large numeric values, the implementation suffers from a state explosion increasing the load and resolution time. As mentioned above, we are aiming at monitoring and maintaining the state of the enactment of open regulated multi-agent systems. To use the implementation of $n\mathcal{C}_+$ in this setting, we would have to add the new agents to the action description file and reload it again. However, the long time that elapses to complete this operation makes unviable the use of the implementation for our purposes and motivated this work.

7 Conclusions and Future Work

In this paper we have introduced a formalism for the management and regulation of concurrent events generated by agents in open MAS. Ours is a rule language in which concurrent events may have a combined effect and may be ignored, forced, expected or sanctioned. The semantics of our formalism relies on transition systems conferring it a well-studied semantics.

We have explored our proposal in this paper by specifying an example of concurrency: soup bowl lifting problem and an example of bank as Electronic Institution.

Although our language is not as expressive as the language of [3] since we cannot post-dict or plan about an action description, our language is not a language for checking properties of a transition system but for specifying its behaviour.

As a proof of concept, an interpreter of $\mathcal{I}$ were implemented in Prolog. As future work, we would like to embed this interpreter in a real MAS and include the distributed management of normative positions introduced in [11].

Acknowledgements – This work was partially funded by the Spanish Education and Science Ministry as part of the projects TIN2006-15662-C02-01 and 2006-5-0I-099 and it was partially done during a stay of the author in Imperial College London. The author wants to thank Marek Sergot for his advise and hospitality. He also thanks Juan-Antonio Rodríguez-Aguilar and Pablo Noriega for their comments and reviews. García-Camino enjoys an I3P grant from the Spanish National Research Council (CSIC).

References

1. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: Norm Oriented Programming of Electronic Institutions. In: Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. (AAMAS'06). (2006)
2. Artikis, A., Kamara, L., Pitt, J., Sergot, M.: A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. In: Declarative Agent Languages and Technologies II. Volume 3476 of LNCS. Springer-Verlag (2005)
3. Sergot, M., Craven, R.: The deontic component of $n\mathcal{C}+$. In: Eighth International Workshop on Deontic Logic in Computer Science (DEON'06). (2006)
4. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Sartor, G., Torroni, P.: Mapping deontic operators to abductive expectations. In: Proceedings of 1st International Symposium on Normative Multiagent Systems (NorMAS 2005), AISB 2005, Hertfordshire, Hatfield, UK (2005)
5. Minsky, N.: Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction, and a Reference Manual). Technical report, Rutgers University (2005)
6. von Wright, G.H.: Norm and Action: A Logical Inquiry. Routledge and Kegan Paul, London (1963)
7. Apt, K.R.: From Logic Programming to Prolog. Prentice-Hall, U.K. (1997)
8. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer-Verlag, New York, U.S.A. (1990)
9. Gelfond, M., Lifschitz, V., Rabinov, A.: What are the limitations of the Situation Calculus? Essays in Honor of Woody Bledsoe (1991) 167–179
10. Esteva, M.: Electronic Institutions: from specification to development. PhD thesis, Universitat Politecnica de Catalunya (2003) Number 19 in IIIA Monograph Series.
11. Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A., Vasconcelos, W.: Distributed Norm Management in Regulated Multi-agent Systems. In: Sixth International Joint Conference on Autonomous Agents and Multiagent Systems. (AAMAS'07). (2007)

Towards a Formalisation of Dynamic Electronic Institutions

Eduard Muntaner-Perich[1], Josep Lluís de la Rosa Esteva[1]

[1] Agents Research Lab, Edifici PIV, Campus de Montilivi, 17071.
Universitat de Girona, Catalonia, Spain
{emuntane, peplluis}@eia.udg.edu

Abstract. This paper presents a formalisation of our Dynamic Electronic Institutions model. In our opinion Dynamic Electronic Institutions arise from the convergence of two research areas: electronic institutions and coalition formation. We believe that these kinds of institutions are potentially important in open-agent system applications because they are well suited to many application domains in which autonomous agents have to collaborate and engage themselves in temporary alliances that require some regulatory measures. This paper presents a brief summary of our previous work on dynamic institutions, introduces the formalisation of our model, explains the process of turning a coalition into a dynamic institution (foundation process), and describes our current and future work.

Keywords: Open Agent Systems, Electronic Institutions, Coalition Formation.

1 Introduction

From a social point of view, it is easy to observe that the interactions between people are often guided by institutions that help and provide us with structures for daily life tasks. Institutions structure incentives in human exchange (political, social, or economic). Somehow we could say that institutions represent the rules of the game in a society or, more formally, are the human-devised constraints that shape human interaction [1].

The idea to use organizational metaphors to model systems was earlier proposed [2, 3]. These approaches suggest structuring the agent society with roles and relationships between agents. But the study of electronic institutions is a relatively recent field (the first approach was [4]). The main idea is simple, and it could be summarized by imagining groups of intelligent, autonomous and heterogeneous agents, which play different roles, and which interact with each other under a set of norms, with the purpose of satisfying individual goals and/or common goals. As a first impression, it could seem that these norms are a negative factor which adds constraints to the system, but in fact they reduce the complexity of the system, making the agents' behaviour more *predictable*. Actually, this is completely true only assuming that agents follow the rules that are created by such norms, and in Open Agent Systems, this is not an assumption to be taken lightly.

Research in Distributed Artificial Intelligence (DAI) has focused on the individual behaviour of agents. But this agent-centred perspective is not useful in complex systems like *open agent systems*, where their components (agents) are not known a priori, can change over time, and can be heterogeneous and exhibit very different behaviours. In these kinds of systems, this vision that is focused on the agent can cause the emergent behaviour of the global system to be chaotic and unexpected. In critical applications this can be a significant problem, and it is evident that it is necessary to introduce regulatory measures which determine what things the agents can do, and what they cannot. It is here where the institutions acquire importance [5].

In Noriega's thesis [4], an abstraction of the notion of institution is introduced for the first time. He is also the first to use the term *agent-mediated electronic institution*, which he describes as: computational environments which allow heterogeneous agents to successfully interact among them, by imposing appropriate restrictions on their behaviours. Continuing and extending the ideas of Noriega's thesis, there is Rodríguez-Aguilar [5] who emphasizes the need for a formal framework which allows to work with general electronic institutions.

From these first approaches to this area, to the actual lines of research, there have been different European research groups working on similar subjects, each one with its particular perspective and approach to the problem. At the moment, many efforts are dedicated to this research area. The proof of this is that in 2003, five PhD theses intimately related to this subject were presented. The theses are: [6, 7, 8, 9, 10].

These different approaches to electronic institutions have demonstrated how organisational approaches are useful in *open agent systems*, but in our opinion, in some application domains that require short to medium-term associations of agents regulated by norms, classical electronic institutions still have several problems and limitations. We have summarized these problems in the following list:

- All the approaches to electronic institutions are based on medium to long-term associations and dependencies between agents. This characteristic is useful in some application domains but it is a significant problem in other domains, where changes in tasks, in information and in resources make temporary associations (regulated by norms) necessary.
- Electronic institutions require a design phase (performed by humans). It is necessary to automate this design phase in order to allow the emergence of electronic institutions (without human intervention) in open agent systems.
- Agents can join and leave institutions, but how do these entrances and exits affect the institutions' norms and objectives? Could these norms and objectives change over time?
- When an institution has fulfilled all its objectives, how can it dissolve itself?

In our opinion, these problems and limitations can be studied and possibly solved with a coalition formation approach to electronic institutions, in order to develop dynamic electronic institutions. This is the main objective of our research.

There is little previous work on dynamic electronic institutions: this idea has just recently been introduced as a challenge for agent-based computing. It first appeared when the term *dynamic electronic institution* appeared in a roadmap for agent technology [11].

Our recent work in this area has involved the development of our dynamic electronic institutions model [12], and some exploratory work in two application

domains: Operations Other Than War simulation [13], and Digital Business Ecosystems [14].

This paper is organized as follows. In section 2 we explain the notion of Dynamic Electronic Institutions and their lifecycle. Next, section 3 illustrates the formalisation of each phase in our model, and our CBR approach to the foundation phase. Before concluding, we discuss some related work in section 4. Finally, section 5 concludes with discussion and future research.

2 Dynamic Electronic Institutions: the Model

We argue that Dynamic Electronic Institutions (DEIs from now on) can be described as follows: emergent associations of intelligent, autonomous and heterogeneous agents, which play different roles, and which are able to adopt a set of regulatory components (norms, missions, coordination protocols, etc) in order to interact with each other, with the aim of satisfying individual goals and/or common goals. These formations are dynamic in the sense that they can be automatically formed, reformed and dissolved, in order to constitute temporary electronic institutions on the fly.

There are several application domains that require short-term agent organisations or alliances, in which DEIs could be applied. Some of them are: Digital Business Ecosystems (we have addressed this topic in [14]), B2B Electronic Commerce, Mobile Ad-Hoc Networks, simulation of Operations Other Than War [13], etc.

In our opinion DEIs should have a lifecycle made up of by three phases: Formation, Foundation and Fulfilment (We call this lifecycle "3F cycle" [12], see Figure 1):

1. *Formation phase*: this is the coalition formation phase. In this stage the objective is for automatic association between agents with the same (or similar) goals to emerge. Other notions such as trust between agents should also be considered as important factors in the coalition formation phase. We are not currently studying coalition formation mechanisms, because we have focused our research on the foundation phase, but in our opinion there are two approaches which introduce some interesting ideas and could be suitable for being used in the formation phase of DEIs: Q-Negotiation (ForEV framework) [15] and the CONOISE project [16].
2. *Foundation phase***:** the process of turning the coalition into a temporary electronic institution. This phase is the real challenge, because the process of turning the coalition into a temporary electronic institution is not a trivial problem. It requires the agents to adopt a set of components that regulate their interactions. This must be an automated process, without any human intervention, so agents must be able to reason and negotiate at a high level.
3. *Fulfilment phase*: this is the dissolution phase. When the institution has fulfilled all its objectives, the association should be broken up. This phase occurs because the association is no longer needed, or because the institution is no longer making a profit. This subject has hardly been explored by current research efforts, and most of the support functionalities need to be developed.

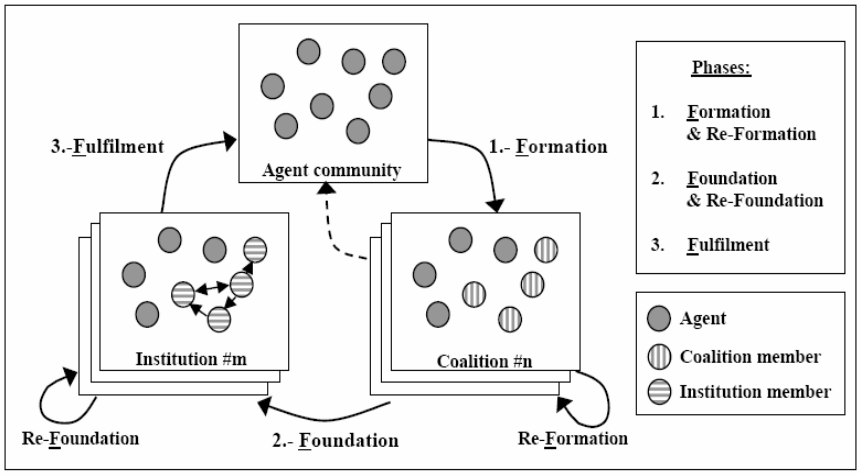


Fig. 1. DEI construction phases (3F cycle)

One of these three phases has been poorly studied in the past: the foundation phase, we are focusing our work on this phase.

3 Formalising the Model

This section presents a formalisation of our model of DEIs. Each phase of the model is formalised, but we are principally focusing our efforts on the foundation phase. In the following subsections we use a basic set theory notation. The term *symbol* refers to a user-defined string (similar to a variable name), and the term *expression* is an algebraic expression (possibly referencing constants, symbols and function calls).

3.1 The Formation Phase

This is the coalition formation phase. As we have said before, we are not currently focusing our research on this phase, but we need to analyze some concepts in order to be able to formalise the foundation phase.

We consider a population P of autonomous and heterogeneous agents (1), consisting of a variable number N of individuals. In this context, a coalition is a subgroup C of agents of P (2). Agents form a coalition because they need to work together to achieve tasks in an environment. Their reason is because mutual profit can be gained from sharing resources and redistributing tasks.

$$P = \{ a_1, a_2, \ldots a_N \} \quad (1)$$

$$C \subseteq P \quad (2)$$

At its simplest the problem can be defined as in [17]: "Given a population P of agents and a list of tasks or goals T, select subgroups of agents S1, S2, S3… of P to address each of the tasks in T". In the general case, this problem is computationally intractable (NP-Hard). This is very easy to understand: given N agents and k tasks, there are k(2N-1) different possible coalitions. The number of coalition configurations (different partitions of the set of agents in coalitions) is of the order $O(N^{(N/2)})$ [18]. Therefore, it is clear that an exhaustive search of the coalition configuration space is not feasible when the number of agents is large. Recent works in distributed artificial intelligence have resulted in distributed algorithms with computational tractability [17].

3.2 The Foundation Phase

We define foundation as the process of turning a coalition into a temporary electronic institution. This phase is a real challenge, and requires the agents to automatically adopt a set of *institutional elements* that regulate their interactions.

Our perspective on this problem is that to construct an institution from zero without human intervention may be too difficult, so we argue that an approach based on using knowledge from previous cases (like Case Based Reasoning, CBR) could be interesting and useful for solving this issue. Presently, we are directing our efforts in this direction.

Therefore, in our system, a stored case (*institution case*) refers to a problem situation and contains a description of a problem, and its solution (the *institutional elements* to be adopted), and a new case (*coalition case*) contains the description of the problem to be solved. Case-based reasoning is a cycle, and there are four phases in the process: *Retrieve*, *Reuse*, *Revise* and *Retain*.

With a CBR approach to the foundation process, when a coalition has been formed and needs to turn itself into an institution, agents should consult their case database in order to find the stored institution's specification that adapts best to the present situation, and should then make the pertinent reforms to the selected specification in order to obtain an institution that works correctly.

The first step in this process is to build a *coalition case CC* from the coalition *C* that has been formed. We consider the *coalition case* as a tuple of different elements (4).

$$C = \{ a_1, a_2, \ldots a_i \} \quad (3)$$

$$CC = < Ty, Tk, Ob, n, div, tr > \quad (4)$$

The components of the *coalition case* are the elements that need to be taken into account when we search the institution that adapts best to the present coalition. These components are:

- *Ty* (types): this component is the set of types of the agents in the coalition. Each type is a *symbol*.

$$Ty = \bigcup_i \{ type(a_i) \} \quad (5)$$

- *Tk* (tasks): this component is the set of tasks of the agents in the coalition. Each task is a *symbol*.

$$Tk = \bigcup_i \{ tasks(a_i) \} \quad (6)$$

- *Ob* (objectives): this component is a set of objectives. These are not the objectives of the coalition (coalition has no objectives; each agent has its own objectives). These are a subgroup of the objectives of all the agents. More specifically, *Ob* is the set of shared objectives, extracted from the intersection of the different sets of objectives. We believe that shared objectives are an important element to take into account when we are searching the institution that best adapts to the present coalition. Each objective is an *expression*.

$$Ob = \bigcap_i \{ objectives(a_i) \} \quad (7)$$

- *n* (number of agents): this component is the number of agents in the coalition.

$$n = |C| \quad (8)$$

- *div* (diversity measure): this component is the diversity within the coalition with respect to the objectives of the agents. This value is measured using an adaptation of Shannon's entropy function:
  Nt = *total number of objectives:* $\sum_i \ |\ objectives(a_i)|$
  K = *number of different types:* $|\ \bigcup_i \{ type(a_i) \}\ |$
  n_a = *number of objectives of type a*
  $P_a = n_a / Nt$

$$div = \frac{-\sum_a^k Pa * \log 2(Pa)}{\log 2(\min(Nt, K))} \quad (9)$$

- *tr* (internal trust): this component is the mean trust value. We calculate it as a double summation: the first one is the sum of all the trust values for an agent with respect to the other agents in the coalition; and the second is the sum of the mean trust of all the agents.

$$tr = \frac{\sum_{i=0}^{n} \left(\frac{\sum_{j=0}^{n} trust(a_i, a_j)}{n} \right)}{n} \quad (10)$$

In our opinion, trust and diversity are important elements to be taken into account, because they capture valuable information of the coalition. We believe that if we are trying to turn a coalition into a temporary institution, diversity and trust among agents should be taken into account to find the institution's specification that adapts best to the present situation. Of course this is a conjecture that needs to be proved. Having this in mind, one of our current efforts aims at developing a framework for DEIs that will help to test it.

When we have the coalition case (*CC*), the next step is to start the CBR process. We need a *previous-institutions base*, which contains the knowledge of the system. In our model this institutions base is called *K* (11). Each case of this base is an *institution case*, *IC* (12), which contains a *CC* and the *institutional elements* (*IE*), that is, the elements that have to be adopted to turn the coalition into a dynamic institution (13).

To initialise the system, an initial set of *institution cases* must be introduced into the case base. Therefore, in the first CBR iterations the coalitions can reuse previous *institution cases*. This set is important, and should capture some general and typical associations among agents in the specific application domain. This process has to be performed by humans, before starting up the system, and then the rest of the processes should be automatic.

$$K = \{ IC_1, IC_2, ..., IC_n \} \quad (11)$$

$$IC = < CC, IE > \quad (12)$$

$$IE = < M, N, F, pr, ont > \quad (13)$$

The *institutional elements IE* (13) are:

- *M* (Missions): sets of specific objectives for each agent, where each objective is an *expression*.
- *N* (Norms): these are the norms to be adopted by the coalition. These can be obligations (*obl*), permissions (*per*), or prohibitions (*pro*). Table 1 shows the internal structure of these norms.
- *F* (Fulfilment Requirements): this component refers to future requirements for the fulfilment phase. It includes:
    - *FC* (Fulfilment Conditions): these are the conditions that allow the execution of the fulfilment process. Each condition is an *expression*.
    - *FN* (Fulfilment Norms): these are the norms (obligations, permissions and prohibitions) that have to be followed during the fulfilment phase. *FN* and *N* have the same internal structure.
- *pr* (Protocol): this is the protocol to be adopted by the coalition. It will steer the communication processes within the dynamic institution.
- *ont* (Ontology): an ontology to be adopted by all agents in the coalition (of course if an agent already has the ontology there is not need to adopt it).

The CBR process compares the present *coalition case* (*CC*) with the *coalition case* included in each *institution case* (*IC*). This process requires some similarity rules. Each component of the *CC* has a specific similarity measure, and there is global similarity that corresponds to a weighted sum of partial similarities (14).

$$Sim = (w_1*SimTy + w_2*SimTk + w_3*SimO + w_4*SimN + w_5*SimDiv + w_6*SimTr) \quad (14)$$

Table 1. Structure of each norm of *N*, with examples

Norm Id	***Condition* (expression)**	***Bearer* (agent type)**	***Type* (obl, per, pro)**	***Mission or Task***	***Deadline***
N_1	Null	Employee	Obl	M_3	<end
N_2	$(a_1>10) \wedge (b_2<5)$	Employer	Obl	M_1	<end

The weights used in the similarity function depend on the specific implementation, and the application domain. They are performance parameters that need to be empirically adjusted. When the *institution case* (*IC*) that best adapts to the *coalition case* (*CC*) is found, an adjustment of the *institutional elements* (*IE*) is required in order to allow the agents of the new coalition to re-use them. This is not a simple process; in fact it can become very complicated, and it depends partially on the specific implementation of the model. The following are some general guidelines:

- The ontology *ont* can be directly adopted by the coalition. We assume that all the agents in the system are using the same ACL (Agent Communication Language) and are able to work with ontologies.
- The protocol *pr* could require a more sophisticated adoption process. If the protocol to be adopted refers to types of agents that do not exist in the new coalition, we need to modify the protocol. There are different possible ways to do it, but we believe that a simple solution could be to maintain a similarity table within *K*, which informs about the similarity between the different types of agents. This way we can replace one type of agent in the protocol with the type of agent that better fits the context requirements.
- The modification of norms *N* is also a hard task. If the bearer of the norm refers to a type of agent that does not exist in the coalition we basically have two options: to eliminate this norm, or to find the type of agent in the coalition that is more similar to the bearer (the same process that has been proposed for the protocol modification can be used). The modification of norms can also consider an internal modification of the norm, more specifically of its *conditions* and *missions*. This kind of modification implies more sophisticated adaptation processes. We can consider the need for more similarity tables (for *missions*, constants and global variables), or another option could be to consider a genetic algorithm for the estimation of these parameters.
- The modification and adoption of the missions *M* and the fulfilment requirements *FR* imply similar changes like those described above for *N*.

Foundation starts by building a *coalition case CC*. The next step is the CBR process that can be expressed as a function (15) that, from a *coalition case CC* and the institutions base *K*, produces the adapted *institutional elements IE*. We need another process that adds these *IE* to the coalition and finally produces the DEI. Therefore, we can conceive the global foundation process as a function that, from a coalition *C* and *K*, produces the DEI (16).

$$CBR: (CC, K) \rightarrow IE \quad \textbf{(15)}$$

$$foundation: (C, K) \rightarrow DEI \quad \textbf{(16)}$$

With this formalisation of the foundation phase we also can consider a *re-foundation* process, which facilitates reconfiguring the dynamic institution when member changes or environment changes occur. Currently, reorganisation within a multi-agent system is a topic that is being actively discussed. Section 4 cites some related work.

Figure 2 shows a diagram of the foundation process.

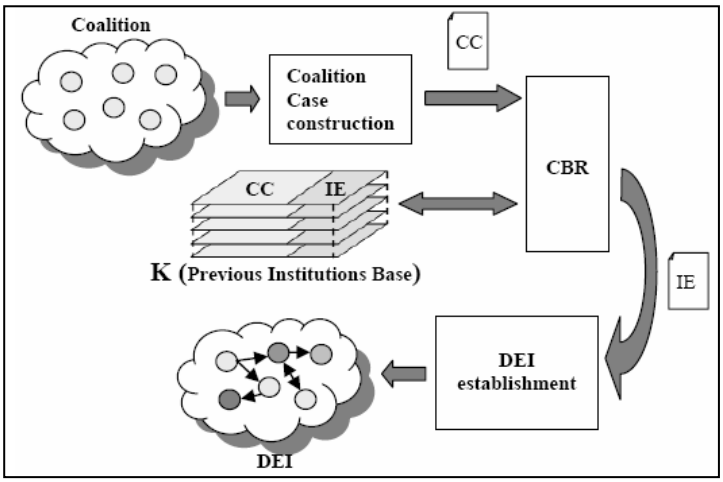


Fig. 2. The Foundation phase

3.3 The Fulfilment Phase

This is the dissolution phase. When the fulfilment conditions *FC* are achieved, the association should be broken up. This phase occurs because the association is no longer needed, or because the institution is no longer making a profit. Within this phase the agents should distribute the profits obtained (following the fulfilment norms *FN*) and store relevant information for future DEIs (a new institution case *IC*).

This subject has hardly been explored by current research efforts, and most of the support functionalities need to be developed. We believe that there are two important steps in this phase: the construction of the new institution case *IC* and the update of the institutions base *K*. The result of this process is an updated institutions base *K'*, that includes the new institution case (17).

$$fulfilment: (DEI, K) \rightarrow K' \quad (17)$$

3.4 Properties of the formalised model

Taking into account the above presented formalisation, we argue that DEIs are:

- Dynamic: DEIs are dynamic in the sense that they can be automatically formed, reformed and dissolved, in order to constitute regulated associations of agents on the fly.
- Automatic: DEIs don't require a design phase performed by humans, although an initial set of *institution cases* (IC) must be introduced into the case base before starting up the system, as it has been said before.
- Temporary: DEIs are conceived as short to medium-term associations. They acquire the fulfilment requirements during the foundation phase.

- Adaptive: CBR provides adaptive solutions by systematic comparison between current coalition and the stored institutions.

4 Related Work

There is little previous work on DEIs. Currently there is a work in progress [19] that is focused on the extension of electronic institutions with autonomic capabilities to allow them to yield a dynamical answer to changing circumstances, through the adaptation of their norms. The authors use a genetic algorithm to learn the best parameters for a population of agents. It is a very interesting approach that could complement our work.

Recently, there is an increasing interest on *organisational self-design*. In [20], the authors affirm that "we must move from an agent-centric view of coordination and control to an organisation-centric one. However, in order to be able to adapt and evolve, this latter will need to coexist with dynamic and (partially) emergent organisation, based on the former". However, although there are many practical applications being developed, there is need for formal theories to describe dynamic organisational structures.

In [21], a general view of the reorganisation problem, within a multi-agent system, is presented. The authors present a reorganisation model where agents have autonomy to change their organisations. Their approach is based on the MOISE$^+$, which is an organisational model for multi-agent systems based on notions like roles, groups, and missions.

An interesting approach to *organisational design* is proposed in [22], where the authors present a distributed algorithm that uses an underlying organisation to guide coalition formation.

Our approach is closely related to the concept of Contractual Agent Societies [23], a metaphor for building open information systems where agents configure themselves automatically through a set of dynamically negotiated social contracts. In [14] we have studied the adoption of institutional components through an electronic contract.

In this article we have not examined Virtual Organisations [15, 16, 24]. This concept is closely related to electronic institutions and coalition formation. In fact, in our opinion, VOs could be described in terms of DEIs, although their architectures and implementations are usually directed to a specific application domain: B2B electronic commerce. We believe that in someway VOs could be considered as a sub-group of DEIs which are more general. In one study [15] the authors work towards the development of an agent-based electronic institution providing a virtual normative environment that assists and regulates the creation and operation of VOs. Their work confirms our idea, because they prove that VOs can be conceived as DEIs. An interesting approach to VOs is proposed in [24], where authors try to formalise VOs and contracts based on commitments.

Finally, there is an approach that studies the dynamic selection of coordination mechanisms among autonomous agents [25]. The authors presented a framework that enables autonomous agents to dynamically select the mechanism they employ in order to coordinate their inter-related activities.

5 Conclusions and Future Work

In this article, we have presented a brief summary of our previous work on DEIs (the general model and the CBR approach) and we have introduced a formalisation of our DEIs model. This formalisation provides us with a preliminary theoretical framework for working with institutions that are dynamic, automatic, temporary, and adaptive.

In our previous works we presented an exploratory work [13] that was focused on the simulation of Operations Other Than War (OOTW) using a first version of our DEIs model. Our first experiments were very simple, but the preliminary results were encouraging. They used a centralized CBR approach on the OOTW domain, and showed that the foundation phase is feasible, and that the DEI lifecycle can be fully implemented.

Currently, we are centring our efforts on the implementation of a framework for DEIs. It will follow our general model and the formalisation presented in this paper. We are using Repast to implement it, and we would like to use it in another application domain: digital business ecosystems (DBEs). We have recently presented a work in this direction [14].

We are using a CBR approach in the foundation phase, but we do not rule out alternative approaches like meta-institutions or genetic algorithms. A Meta-Institution could provide general modules (norms, ontologies, protocols, etc.), which have to be instantiated in order to build specific DEIs.

At this moment, we are involved in the ONE Project (Open Negotiation Environment [26]), which tries to enrich digital business ecosystems with an open, decentralised negotiation environment. As we have said before, we would like to use our DEIs model to enable these digital business ecosystems.

Acknowledgments. This research was partially funded by EU project Nº 34744 ONE: Open Negotiation Environment, FP6-2005-IST-5, ICT-for Networked Businesses.

References

1. North, D. C.: Economics and Cognitive Science. Economic History 9612002, Economics Working Paper Archive at WUSTL (Washington University in St. Louis), (1996)
2. Pattison, H. E., Corkill, D. D., and Lesser, V. R.: Distributed Artificial Intelligence, chapter Instantiating Descriptions of Organizational Structures, pages 59-96. Pitman Publ. (1987)
3. Werner, E.: Distributed Artificial Intelligence, chapter Cooperating Agents: A Unified Theory of Communication and Social Structure, pages 3-36. Pitman Publ. (1987)
4. Noriega, P.: Agent Mediated Auctions. The Fishmarket Metaphor. Ph.D.Thesis, Artificial Intelligence Research Institute (IIIA), Universitat Autònoma de Barcelona (1997)
5. Rodríguez-Aguilar, J. A.: On the design and construction of Agent-mediated Electronic Institutions. Ph.D. thesis, Artificial Intelligence Research Institute (IIIA), Universitat Autònoma de Barcelona (2001)
6. Esteva, M.: Electronic Institutions: From specification to development. PhD thesis, Artificial Intelligence Research Institute (IIIA), Universitat Politècnica de Catalunya (2003)

7. Dignum, V.: A model for organizational interaction. Based on Agents, Founded in Logic. Ph.D. Thesis. Dutch Research School for Information and Knowledge Systems, Utrecht University (2003)
8. Fornara, N.: Interaction and communication among autonomous agents in multiagent systems, Ph.D. Thesis, Università della Svizzera italiana, Facoltà di Scienze della Comunicazione (2003)
9. López y López, F.: Social power and norms. Impact on Agent Behaviour. Ph.D. Thesis, University of Southampton, Department of Electronics and Computer Science (2003)
10. Vázquez-Salceda, J.: The role of Norms and Electronic Institutions in Multi-Agent Systems applied to complex domains. The HARMONIA framework. PhD thesis, Universitat Politècnica de Catalunya, Dept. Llenguatges i Sistemes Informàtics. Artificial Intelligence Dissertation Award, ECCAI, (2003)
11. Luck, M., McBurney, P., Preist, C.: Agent Technology: Enabling Next generation Computing. A Roadmap for Agent Based Computing. AgentLink II (2003)
12. Muntaner-Perich, E, de la Rosa, J.Ll.: Towards Dynamic Electronic Institutions: from agent coalitions to agent institutions. Published in LNCS, Vol. 3825: Innovative Concepts for Autonomic and Agent-Based Systems, pp. 109-121, Springer Verlag, (2006)
13. Muntaner-Perich, E., de la Rosa, J.Ll, Carrillo, C., Delfín, S., Moreno, A.: Dynamic Electronic Institutions for Humanitarian Aid Simulation. Publ. in Frontiers in AI and Applications - AI Research & Development, vol. 146, pp. 239-246, IOS Press, (2006)
14. Muntaner-Perich, E., de la Rosa, J.Ll.: Using Dynamic Electronic Institutions to Enable Digital Business Ecosystems. COIN'06 Workshop: Coordination, Organization, Institutions and Norms in Agent Systems, ECAI 2006. Riva del Garda, Italy, Aug 28th-Sept 1st (2006)
15. Rocha, A. P., Lopes Cardoso, H., Oliveira, E.: Contributions to an Electronic Institution supporting Virtual Enterprises' life cycle. In G. Putnik e M. M. Cunha (eds.), Virtual Enterprise Integration: Technological and Organizational Perspectives, Idea Group Inc, ISBN 1-59140-406-1, in press. (2005)
16. Dang, V. D.: Coalition Formation and Operation in Virtual Organisations, PhD Thesis. School of Electronics and Computer Science, University of Southampton, (2004)
17. Klusch, M., Gerber, A.: Dynamic coalition formation among rational agents. IEEE Intelligent Systems, 17(3):42-47 (2002)
18. Shehory, O.: Coalition Formation: Towards Feasible Solutions. Proc. of International Central and Eastern European Conference on MAS. Prague, Springer-Verlag: 4-6, (2003).
19. Bou, E., López-Sánchez, M., and Rodriguez-Aguilar, J.A.: Norm adaptation of autonomic electronic institutions with multiple goals. International Transactions on Systems Science and Applications, 1(3):227–238, (2006)
20. Sichman, J. S., Dignum, V., Castelfranchi, C.: Agents´ organizations: a concise overview In: Journal of the Brazilian Computer Society, 11(1), pages 3-8 (2005)
21. Hübner, J. F., Sichman, J. S., Boissier, O.: Using the Moise+ for a Cooperative Framework of MAS Reorganisation. SBIA 2004: 506-515 (2004)
22. Horling, B. and Lesser, V.: A Survey of Multi-Agent Organizational Paradigms. The Knowledge Engineering Review 19(4):281-316 (2005)
23. Dellarocas, C.: Contractual Agent Societies: Negotiated shared context and social control in open multi-agent systems. Proc. WS on Norms and Institutions in Multi-Agent Systems, Autonomous Agents-2000, Barcelona (2000)
24. Udupi, Y. B. and Singh, M. P.: Contract Enactment in Virtual Organizations: A Commitment-Based Approach. Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), Boston, AAAI Press, pages 722–727 (2006)
25. Excelente-Toledo, C. B. and Jennings, N. R.: The dynamic selection of coordination mechanisms. Autonomous Agents and Multi-Agent Systems 9(1-2) pp. 55-85 (2004)
26. ONE Project.: EU project Nº 34744, ONE: Open Negotiation Environment. Website: http://one-project.eu, (2007)

A Contract Model for Electronic Institutions

Henrique Lopes Cardoso, Eugénio Oliveira

LIACC – NIAD&R, Faculty of Engineering, University of Porto
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
{hlc, eco}@fe.up.pt

Abstract. Electronic institutions are software frameworks integrating normative environments where agents interact to create mutual commitments. Contracts are formalizations of business commitments among groups of agents, and comprise a set of applicable norms. An electronic institution acts as a trusted third-party that monitors contract compliance, by integrating in its normative environment the contractual norms, which are applicable to the set of contractual partners. In this paper we present and explore a contract model that facilitates contract establishment by taking advantage of an institutional normative background. Furthermore, the model is flexible enough to enable the expansion of the underlying normative framework, making it applicable to a wide range of contracting situations.

1 Introduction

Research on norms and multi-agent systems has grown the Electronic Institution (EI) concept as the basis for the development of appropriate normative environments. Such environments are created to establish some kind of social order [4] that allows successful interactions among heterogeneous and autonomous entities.

As with any recent discipline, however, differences exist between the conceptual views of the "institutional environment". Some authors [1] advocate in favor of a restrictive "rules of the game" approach, where the EI fixes what agents are permitted and forbidden to do and under what circumstances. In this case norms are a set of interaction conventions that agents are willing to conform to. Other researchers [2] take a different standpoint, considering the institution as an external entity that ascribes institutional powers and normative positions, while admitting norm violations by prescribing appropriate sanctions. Others still [9] focus on the creation of institutional reality from speech acts, regarding an agent communication language as a set of conventions to act on a fragment of that reality.

A common element in each of these approaches is the *norm*, which enables us to control the environment, making it more stable and predictable. Arguably, one of the main distinguishing factors among researchers using norms in institutions is the level of control one has over agents' autonomy.

Our own view of electronic institutions (as initiated in [14] and developed in [13]) has got two main features that motivate the present paper. Firstly, the institution includes a set of services that are meant to assist (not only regulate) agent interaction

and the creation of new normative relationships. This means we do not take the environment as static from a normative point of view (as seems to be the case in [1]). New commitments may be established among agents, through contract negotiation (as also noted by [3]); the resulting contracts comprise a set of applicable norms. Additionally, part of the aforementioned assistance is achieved by enriching the institutional environment with a supportive normative framework. This will allow contracts to be underspecified, relying on default norms that compose the institution's normative environment where the contract will be supervised.

In this paper we present and explore the definition of a contract model that takes advantage of an institutional normative framework. The model is flexible enough to encompass contracts of varying degrees of complexity. A contract is established with support of the normative background and relying on a model of institutional reality.

The rest of the paper is organized as follows. Section 2 briefly describes the institutional environment that supports the contract model presented in this paper. Section 3 addresses the contract model itself, including its motivation and detailing its constituent parts. The model tries to take advantage of the underlying environment while at the same time enabling the expansion of the normative framework. Section 4 explains contract handling within our electronic institution framework, focusing on the representation of contracts in a computational way. Finally, section 5 concludes by highlighting the main features of our approach.

2 Institutional Environment

The notion of multi-agent systems assumes the existence of a common environment, where agent interactions take place. Recently more attention is being given to the environment as a first-class entity [17]. In the case of electronic institutions, they provide an environment whose main task is to support governed interaction by maintaining the normative state of the system, embracing the norms applicable to each of the interacting agents.

In order to accomplish such task, in our approach [13] the EI is responsible for recording events that concern *institutional reality*. This reality is partially constructed by attributing institutional semantics to agent interactions.

As mentioned before, we seek to have an EI environment with a supportive normative framework. For this, norms are organized in a hierarchical structure, allowing for norm inheritance as "default rules" [5].

2.1 Elements of Institutional Reality

The institutional environment embraces a set of events composing a reality based on which the normative state of the system is maintained. Norm compliance is monitored consistently with those events, which can be grouped according to their source:

- *Agent-originated events*: in our approach, norm compliance detection is based on the assumption that it is in the best interest of agents to publicize their abidance to commitments. They do so by provoking the achievement of corresponding

institutional facts (as described in [13]), which represent an institutional recognition of action execution.

- *Environment events*: norms prescribe *obligations* when certain situations arise. In order to monitor norm compliance, the institutional environment applies a set of rules that obtain certain elements of institutional reality, including the *fulfillment* and *violation* of obligations. While fulfillment acknowledgement is based on institutional facts, violations are detected by keeping track of *time*, using appropriate time ticks. Both norms and rules may use institutional facts as input. Rules also allow obtaining new institutional facts from old ones.

These events are the elements of institutional reality summarized in Table 1.

Table 1. Elements of institutional reality

Element	*Structure*
institutional fact	ifact(<*IFact*>, <*Timestamp*>)
obligation	obligation(<*Agent*>, <*IFact*>, <*Deadline*>)
fulfillment	fulfilled(<*Agent*>, <*IFact*>, <*Timestamp*>)
violation	violated(<*Agent*>, <*IFact*>, <*Timestamp*>)
time	time(<*Timestamp*>)

Because of the normative framework's organization (as explained in the next section), elements of institutional reality are *contextualized*, that is, they report to a certain context defined inside the institutional background.

Our norm definition is equivalent to the notion of conditional obligation with deadline found in [8]. In particular, an *Ifact* (an atomic formula based on a predefined ontology) as included in an obligation comprises a state of affairs that should be brought about, the absence of which is the envisaged agent's responsibility; intuitively, an achievement of such state of affairs before the deadline fulfills the obligation. The *Deadline* indicates a temporal reference at which an unfulfilled obligation will be considered as violated. Fulfilled or violated obligations will no longer be in effect. Monitoring rules capture these semantics, by defining causal links (as described in [7]) between achievements and fulfillments, and between deadlines and violations.

There is a separation of concerns in norm definition and norm monitoring. The latter is seen as a context-independent activity. Also, the detection of norm (or, strictly speaking, obligation) fulfillment or violation is distinguished from repair measures, which may again be context-dependent (e.g. through contrary-to-duty obligations). This approach differs from [16], where norms include specific violation conditions, detection and repair measures.

2.2 Normative Framework

Our view of the EI concept [13] considers the institution as an environment enforcing a set of institutional norms, but also allowing agents to create mutual commitments by voluntarily adhering to a set of norms that make those commitments explicit. The EI will act as a trusted third-party that receives contracts to be monitored and enforced.

Furthermore, with the intent of facilitating contract formation, we approach the normative framework using a hierarchical approach, enabling the adoption of contract law concepts such as the notion of "default rules" [5]. These enable contracts to be underspecified, relying instead on an established normative background. The grouping of predefined norms through appropriate contexts also mimics the real-world organization of legislations applicable to specific activities. These norms will be imposed when the activity they regulate is adhered to by agents.

Our approach consists of organizing norms through *contexts*. Each contractual relationship is translated into a new context specifying a set of norms while inheriting others from the context within which it is raised. The top-level context is the EI itself.

A context definition includes the information presented in Table 2. The *super-context* (which may often be the EI itself) indicates where the current context may inherit norms from, while the *context type* dictates what kinds of norms are applicable (those that govern this type of relationship).

Table 2. Context definition information

Component	*Description*
super-context	the context within which this context was created
type	the type of context
id	the context identifier
when	the starting date of the underlying contract
who	the participants of the underlying contract

The components described in the table are meant to provide structure to our normative framework. It is the normative environment's responsibility to use this structured context representation in order to find applicable norms in each situation.

The specificity of norms will require further information regarding the contract to which they apply. For this, we consider the explicit separate definition of contextual-information, which will be dependent on the type of context at hand. For instance, in a simple purchase contract, the delivery and payment obligations will need information about who are the vendor and customer, what item is being sold and for what price.

3 Contract Model

This section will provide a description of our proposed contract model. We will start by providing the main assumptions that guided the approach, and proceed with the details of each contract piece. The figures illustrating contract sections were obtained using Altova® XMLSpy®.

3.1 Guidelines

When devising our contract model, we considered the main principles that should guide this definition. On one hand, as stated before we wanted a model that could take advantage of an established normative environment; therefore, each contract should

be obtainable with little effort, and with as few information as possible. On the other hand, we also wanted to make the contract model as expansible as possible, allowing for the inclusion of non-predefined information and norms, while still keeping it processible by the EI environment. This requirement will allow us to apply the EI platform to different business domains.

The contract model should therefore allow us to:

- Include information necessary for context creation, and additionally any contract-type-dependent information to be used by institutionally defined norms.
- Add contract-specific details that are meant to override default institutional norms, e.g. by defining contract-specific norms.
- Expand the predicted contract scenarios by enriching the environment's rules for institutional fact generation.

The next sections describe how each of these purposes is handled.

3.2 Contract Header

Although, in general, a contract may include rules and norms, in the extreme case a contract that is to be monitored by the EI may be composed only of its header. Everything else (including the applicable norms) may be inherited from the EI. This minimalist case is illustrated in Figure 1, where dotted lines indicate optional components that we will refer to later. The rounded rectangle with ellipses is a compositor indicating a sequence of components.

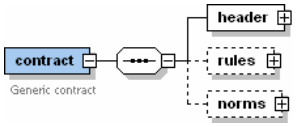


Fig. 1. Generic contract

The contract header (Figure 2) includes mandatory information that is needed for context definition, namely: the contract *id*, the creation date (*when*), and the participants' identification (*who*). The *type* of contract is optional; if not defined, a generic context type will be assumed. The *super-context* is also optional; if omitted, the general EI context is assumed.

Depending on the contract type, additional information may need to be provided. This information can be included in a frame-based way: each peace of *contractual-info* (Figure 3) has a name and a set of slots (name/value pairs).

Finally, each contract may indicate the state-of-affairs according to which the contract will be terminated. The structure of *ending-situation* is analogous to the situation component of a norm definition (as described in the following section).

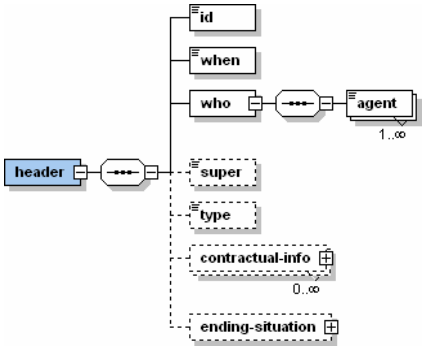


Fig. 2. Contract header

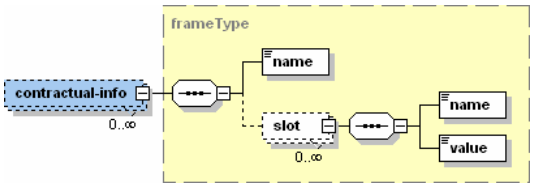


Fig. 3. Contract-type-dependent contractual-info

3.3 Adding Contract-Specific Norms

One way of escaping the default institutional normative setting is by defining norms that are to be applied to a particular contract instance. This is irrelevant of the contract having or not a type as indicated in its heading. A contract of a certain type will inherit institutional norms that are applicable to that type of contract as long as no other contract-specific norms override them. A contract with no type at all will need its norms to be defined in the contract instance.

In our conceptualization, a *norm* prescribes obligation(s) when a certain state-of-affairs is verified (Figure 4). A name is given for norm identification purposes.

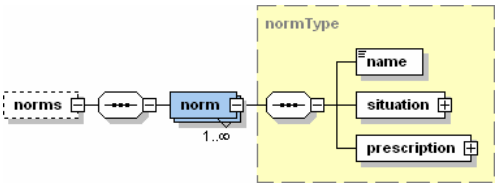


Fig. 4. Contractual norm

The *situation* may be described by institutional reality elements (except obligations) and access contractual-info. Figure 5 includes a choice compositor for situation elements, which may be combined by the logical connectives *and*, *or*, and *not*. Relational conditions may be included to compare numeric values.

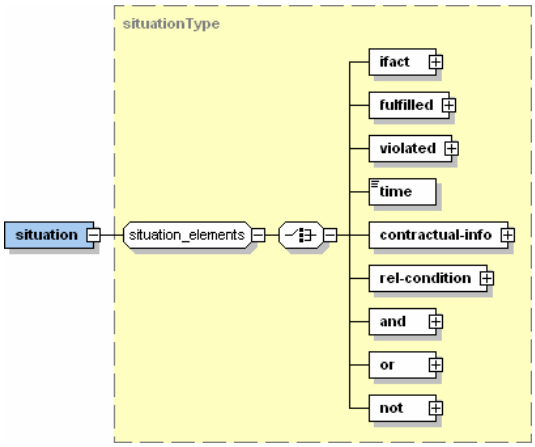


Fig. 5. Situation assessment

The situation elements *ifact*, *fulfilled* and *violated* match the corresponding institutional reality elements (see Figure 6 and Table 1), as does *time*.

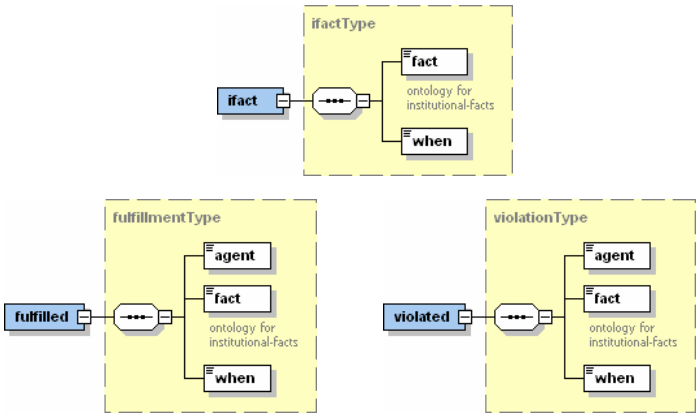


Fig. 6. Situation elements from institutional reality

The *prescription* of norms indicates *obligations* (Figure 7), which have a similar structure to the corresponding institutional reality element.

The usage of institutional reality elements and contractual-info inside norms is allowed to use variable bindings inside appropriate patterns (e.g. within facts and according to the employed ontology), such that they can be referred to in other norm

components. In the future our schema definition will evolve to make this more precise (getting input from other XML rule languages such as RuleML or JessML). For now, we simply assume that variables may appear anywhere inside the mentioned elements and starting with a question mark ('?'), which is Jess's syntax [10].

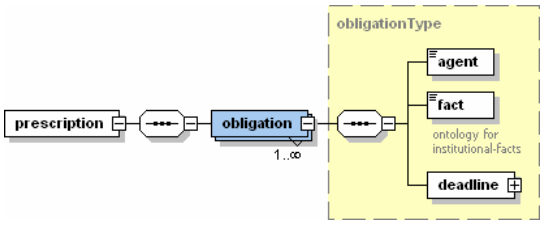


Fig. 7. Obligation prescription

When including norms in a contract-specific way, the normative environment will consider as applicable the most specific norms, that is, those with a narrower scope. This allows a contract to override predefined norms from a super-context (if specified). The same approach is taken when defining a contract-specific ending situation (in the contract header), which may also be predefined for certain context types.

3.4 Expanding the Creation of Institutional Facts

Following a "counts-as" approach (defining "constitutive rules" [15] or "empowerments" [12]), we attribute institutional semantics to agent illocutions. That is, institutional facts, which are part of institutional reality, are created from these illocutions. This process takes place at an institutional context.

In order to assure the applicability of our environment to different contracting situations, we also included the possibility of iterating through institutional facts (although this is also the case in [15], we take a slightly different perspective [13]). That is, certain contractual situations may consider that certain institutional facts (as recognized by the EI) are sufficient to infer a new institutional fact. The rules that allow these inferences to take place are context-dependent and may be specified in a contract-instance basis (see Figure 8). A rule name is given for identification purposes.

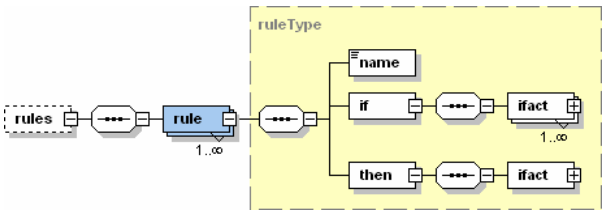


Fig. 8. Rule definition for institutional facts

We consider the iterative generation of institutional facts as context-dependent because it allows contract fulfillment to be adjusted by matters of trust between contractual partners or due to business specificities. Thus, it may be the case that only in specific contractual relationships some institutional fact(s) count as another one.

This approach also enhances the expansibility of the system, not restricting norm definition to the institutional fact ontology defined in the preexistent fact-generating rules. It may be the case that a contract defines new institutional facts through these rules and also incorporates norms that make use of them.

4 Contract Handling in the Electronic Institution

The contract model described in the previous section comprises an XML schema from which contracts are drafted in the contract negotiation phase. The EI provides a negotiation mediation service for this purpose. After this, the negotiation mediator hands over the contract to a notary service, who collects signatures from the involved agents. After this process is completed, the notary requests the EI to include the contract in its normative environment. The contractual norms will then be part of the normative state of the system, and the EI will be responsible for maintaining this state by monitoring the compliance of the involved agents. Figure 9 illustrates this process.

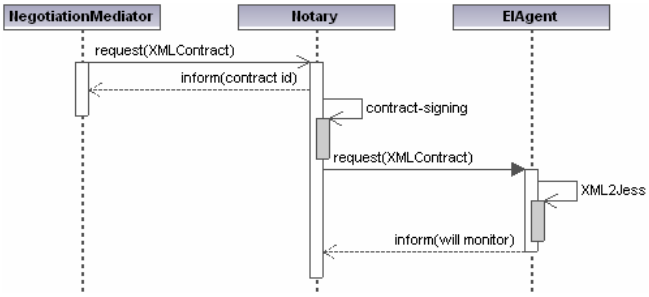


Fig. 9. Contract handling

4.1 From XML to a Computational Contract Representation

In order to achieve a computational normative environment, a declarative language was chosen for norm representation and processing. Furthermore, in order to facilitate communication with the rest of the agents, the EI includes an agent personifying the institution itself and its normative environment. This agent includes an instance of a Jess rule-engine [10], which is responsible for maintaining the normative state of the system and to apply a set of procedures concerning the system's operation.

Hence, in order to allow its processing by the normative environment, the XML contract undergoes a process of transformation into appropriate Jess constructs. (see Figure 10). The Jess language includes a set of frame-like constructs.

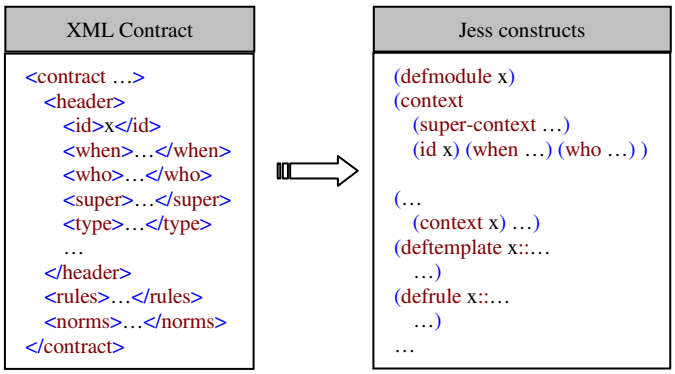


Fig. 10. From XML to Jess

The generated Jess code will be added to the Jess engine, and comprises information regarding the contract creation (which includes a Jess *module* definition and a context construct), optional contextual-info (and associated Jess *template* definitions), and applicable rules and norms (defined as Jess *rules*).

A rule-based approach to norm representation and monitoring is also pursued in [11]. However, those authors seem to implement in a backward-chaining logic program the semantics of a forward-chaining production system. We follow a more intuitive approach by employing a forward-chaining shell.

4.2 Norm Monitoring and Inheritance

The module definition and the structured context representation (using *super-context* relations), are the cornerstones for enabling norm inheritance. Norms are defined inside the module representing the contract's context (that is what the "*x::*" after *defrule* stands for, where *x* is the module/context name). When applying rules, the Jess engine looks at a focus stack containing modules where to search rules for firing. When no rules are ready to fire in the module at the top of the stack, that module is popped and the next one becomes the focus module.

Exploiting this mechanism, we implemented rules that manage the focus stack and thereby enable the application of the most specific norms in the first place. The event that triggers these rules is the occurrence of a new institutional reality element (IRE), which as explained before pertains to a certain context.

The Jess engine will therefore be guided to look for a module where there is an applicable rule taking the IRE as input. It will start at the IRE's module, and go up one level until the top (*main*) module is reached or the IRE is processed.

This initial exploitation of Jess's features enabled us to start building a proof-of-concept regarding our approach to norm inheritance in a hierarchical normative structure. Further refinements will allow us to configure the system concerning monitoring responsiveness and the integration of social extensions like reputation mechanisms.

5 Conclusions

The EI concept has been approached from different perspectives. Considering the increasing importance of multi-agent system environments [17], the EI can be seen as an interaction-mediation infrastructure maintaining the normative state of the system.

One of the most important principles of our approach is the assumption of a non-static normative environment; this means that we depart from a more conservative view of norms seen as a set of preexistent interaction conventions that agents are willing to comply with (as in the *adscription* approach of [1]). We pursue an EI that provides a supportive normative framework whose main purpose is to facilitate the establishment of further commitments among a group of contracting agents.

The possibility of having an underlying normative framework, from which norms may be inherited, is a distinguishing feature of our approach, as is the "loose coupling" between norms and contrary-to-duties. Also, the institution includes norm monitoring policies that span all created contracts. This is in contrast with other approaches, namely [16], where these policies and repair measures are spread among the norms themselves.

The hierarchical organization of norms takes inspiration in the real-world. The most useful case for "default rules" [5] is in defining contrary-to-duty situations, which typically should be not likely to occur. For this reason, such situations are not dealt with in each contractual agreement, and parties usually recur to law systems that include default procedures [6].

In this paper we presented our approach towards the definition of a contract model that can exploit such an environment. The model was devised taking into account two aims: it should be easy to compose a new contract, by taking advantage of an institutional normative background; and it should be possible to improve on the EI's environment in order to make it applicable to different business domains.

We are confident that we have met both these goals. In our model, a minimalist contract may be limited to header information including the contract participants and contractual-info that describes the negotiated objects. On the other hand, a complex unnoticed contractual relationship may be defined using our contract model, by exploiting the whole structure including contract-specific norms and institutional fact generating rules. The next steps of this work include exploring the developed contract model through different contracting scenarios.

Acknowledgments. This project is supported by FCT (Fundação para a Ciência e a Tecnologia) under Project POSC/EIA/57672/2004. Henrique Lopes Cardoso enjoys the FCT grant SFRH/BD/29773/2006.

References

[1] J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar and C. Sierra, *Environment engineering for multiagent systems*, Engineering Applications of Artificial Intelligence, 18 (2005), pp. 191-204.

[2] A. Artikis, J. Pitt and M. Sergot, *Animated specifications of computational societies*, in C. Castelfranchi and W. L. Johnson, eds., *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Association for Computing Machinery, New York, NY 10036-5701, United States, Bologna, Italy, 2002, pp. 1053-1062.

[3] G. Boella and L. van der Torre, *Contracts as Legal Institutions in Organizations of Autonomous Agents*, in N. R. Jennings, C. Sierra, L. Sonenberg and M. Tambe, eds., *Third International Joint Conference on Autonomous Agents & Multi Agent Systems*, ACM Press, New York, NY, United States, 2004, pp. 948-955.

[4] C. Castelfranchi, *Engineering Social Order*, in A. Omicini, R. Tolksdorf and F. Zambonelli, eds., *Engineering Societies in the Agents World*, Springer, Berlin, Germany, 2000, pp. 1-18.

[5] R. Craswell, *Contract Law: General Theories*, in B. Bouckaert and G. De Geest, eds., *Encyclopedia of Law and Economics*, Edward Elgar, Cheltenham, 2000, pp. 1-24.

[6] A. Daskalopulu and T. Maibaum, *Towards Electronic Contract Performance*, *12th International Conference and Workshop on Database and Expert Systems Applications*, IEEE C. S. Press, 2001, pp. 771-777.

[7] F. Dignum, J. Broersen, V. Dignum and J.-J. Meyer, *Meeting the deadline: Why, when and how*, *3rd Conference on Formal Aspects of Agent-Based Systems (FAABS III)*, Springer Verlag, Heidelberg, D-69121, Germany, Greenbelt, MD, United States, 2004, pp. 30-40.

[8] V. Dignum, J.-J. C. Meyer, F. Dignum and H. Weigand, *Formal Specification of Interaction in Agent Societies*, in M. Hinchey, J. Rash, W. Truszkowski, C. Rouff and D. Gordon-Spears, eds., *Formal Approaches to Agent-Based Systems*, Springer Verlag, Heidelberg, Germany, Greenbelt, MD, United States, 2003, pp. 37-52.

[9] N. Fornara, F. Viganò and M. Colombetti, *Agent Communication and Institutional Reality*, in R. M. van Eijk, M.-P. Huget and F. Dignum, eds., *Agent Communication: International Workshop on Agent Communication*, Springer Verlag, Heidelberg, D-69121, Germany, New York, NY, United States, 2005, pp. 1-17.

[10] E. Friedman-Hill, *Jess in Action*, Manning Publications Co., 2003.

[11] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra and W. Vasconcelos, *Norm-Oriented Programming of Electronic Institutions: A Rule-based Approach*, *Coordination, Organization, Institutions and Norms in agent systems (COIN'06)*, Hakodate, Japan, 2006.

[12] A. Jones and M. Sergot, *A Formal Characterisation of Institutionalised Power*, Logic Journal of the IGPL, 4 (1996), pp. 427-443.

[13] H. Lopes Cardoso and E. Oliveira, *Electronic Institutions for B2B: Dynamic Normative Environments*, Artificial Intelligence and Law (in press).

[14] H. Lopes Cardoso and E. Oliveira, *Virtual Enterprise Normative Framework within Electronic Institutions*, in M.-P. Gleizes, A. Omicini and F. Zambonelli, eds., *Engineering Societies in the Agents World V*, Springer Verlag, Heidelberg, D-69121, Germany, Toulouse, France, 2005, pp. 14-32.

[15] J. R. Searle, *The Construction of Social Reality*, Free Press, New York, 1995.

[16] J. Vázquez-Salceda, H. Aldewereld and F. Dignum, *Implementing norms in multiagent systems*, in G. Lindemann, J. Denzinger, I. J. Timm and R. Unland, eds., *Multiagent System Technologies*, Springer Verlag, Heidelberg, D-69121, Germany, Erfurt, Germany, 2004, pp. 313-327.

[17] D. Weyns, A. Omicini and J. Odell, *Environment as a first class abstraction in multiagent systems*, Journal of Autonomous Agents and Multi-Agent Systems, 14 (2007), pp. 5-30.

Coordination in Disaster Management and Response: a Unified Approach

Myriam Abramson, William Chao, Joseph Macker, Ranjeev Mittu

Naval Research Laboratory
4555 Overlook Ave., Washington DC 20375, USA
{myriam.abramson,william.chao,joseph.macker,ranjeev.mittu}@nrl.navy.mil

Abstract. Natural, technological and man-made disasters are typically followed by chaos that results from an inadequate overall response. Three separate levels of coordination are addressed in the mitigation and preparedness phase of disaster management where environmental conditions are slowly changing: (1) communication and transportation infrastructure, (2) monitoring and assessment tools, (3) collaborative tools and services for information sharing. However, the nature of emergencies is to be unpredictable. Toward that end, a fourth level of coordination – distributed resource/role allocation algorithms of first responders, mobile workers, aid supplies and victims – addresses the dynamic environmental conditions of the response phase during an emergency. A tiered peer-to-peer system architecture could combine those different levels of coordination to address the changing needs of disaster management. We describe in this paper the architecture of an agent-based coordination decision support system for disaster management and response and the applicable coordination algorithms including a novel, self-organized and semi-centralized algorithm for team formation.

1 Introduction

Large scale disasters are characterized by catastrophic destruction of infrastructure (e.g., transportation, supply chain, environmental, communication, etc). The lack of coordination characterizes such disasters. While preparedness is the best response to emergencies, a multiagent-based approach to coordination decision support systems (CDSS) can play an important role in disaster management and response (DM&R) in shaping decentralized decision-making on a large scale. However, the diverse aspects of coordination make it difficult to find a unified approach for continuous control. Coordination is at best defined as an emergent property from local interactions in the pursuit of multiple goals and can be either cooperative or competitive. Finding a unified approach is a key problem in disaster management because a cooperative approach in the preparedness phase has to be complemented with a competitive approach in the response phase due to life-threatening situations requiring fast and reactive solutions. A disaster management task is specified by the tuple $\{P, T, A, S\}$ where P is the set

of plans, T the set of tasks or incidents, A the set of agents, volunteers, first responders, and coordinators, S the set of sensors, static or mobile, and where $A \subseteq S$. The problem consists of matching the needs of T with the resources of A in a decentralized and concurrent fashion to accomplish goals defined by P.

This paper is organized as follows. First, we explain the agent-based CDSS framework in Sect. 2 and motivate a tiered peer-to-peer (P2P) coordination architecture for integrating the different coordination dimensions of DM&R. In Sect. 3 we introduce a basic role allocation algorithm for heterogeneous agents suitable in disaster management response. In Sect. 4, a self-organized, semi-centralized coordination algorithm is introduced in support of the architecture proposed. An empirical evaluation follows in Sect. 5 on a canonical fire/rescue scenario to illlustrate the relative merits of the coordination algorithms. Finally, Sections 6 and 7 conclude with related work.

2 Agent-Based CDSS

Recent technological advances in communication and processing power, enabling sensor networks and personal digital assistants, have made possible the self-organization of mobile agents (robots or people) and geo-localized decision support. The complexity of decentralized decision-making is tamed by delegating certain management tasks to proxy agents [1]. Coordination is a pervasive management task that helps reduce interference in role assignments and enhance information sharing. The degree of consensus to obtain before making a decision can be arbitrarily set. The lower the degree of consensus, the more flexible the agents are in reacting to outside events and making timely decisions but the more negative interactions can occur. Assuming rational communicating and trusting agents reduces the degree of consensus overhead required in coordination tasks because the agents are likely to reach the same conclusions given the same information.

Current collaborative web-based tools have essentially a fixed client/server approach because of the relatively stable nature of internet routing. Coordination is achieved through the server as a synchronizing blackboard passively mediating the interactions of intelligent agents as clients. P2P approaches, such as JXTA [2], de-emphasize the role of the server as passive synchronizer but the role of mediator is taken up actively by "rendez-vous" peers and "relays." Peers discover each other through advertisements propagated by relays and rendez-vous peers. This suggests a flexible, semi-centralized coordination architecture for complex tasks such as DM&R where the preparedness and information sharing architecture can seamlessly adapt to rapidly changing conditions and communication infrastructure (Fig. 1). In this framework, coordination at the network layer, whereby a host is chosen to act as relay for propagating messages through the network, maps with a coordinator role at the application layer.

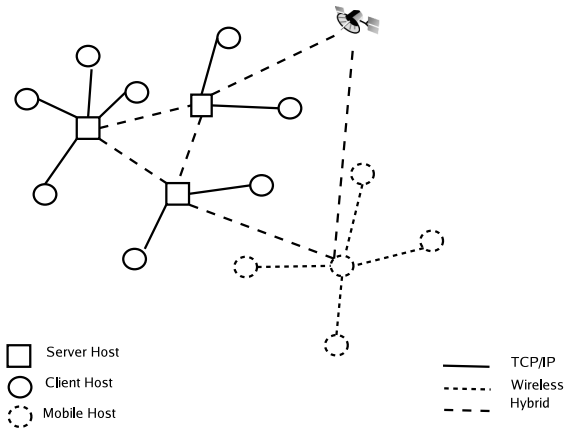


Fig. 1. Semi-Centralized Coordination Architecture

3 Role Allocation for Heterogeneous Agents

One of the key coordination problem in disaster management is the heterogeneity of the players involved. Roles provide a convenient a priori decomposition of a task and are a key coordination tool [3]. Roles can be viewed either as fixed slots in a team structure that are filled by agents or part of an agent's behavior repertoire in its relationships with other agents that can determine the structure of a team. The decision complexity for the role allocation of N agents to p tasks is $O(p^T)$ where T is the number of teams of size t that can be selected from N agents:

$$T = \begin{cases} \binom{N}{t} & case\,1: \; homogeneous\,agents \\ \binom{N+t+1}{t} & case\,2: \; heterogeneous\,agents \end{cases} \tag{1}$$

t is the number of roles in a team which might not correspond to the number of agents N. In the first case, $\sum_i^T t_i \leq N$, agents have distinct, mutually-exclusive roles. In the second case, agents fill a number of non mutually-exclusive roles (i.e. an agent can perform a number of roles in a team). The complexity in role allocation scales up with heterogeneous agents where the mapping of agents to roles is one-to-many. The basic agent loop for role allocation in distributed cooperative systems is described in Alg. 1.

The Greedy Set Cover algorithm is a basic matching algorithm for generalized role allocation of heterogeneous agents running in polynomial time. This is an approximate matching algorithm [4] that finds the minimum set cover for a list of resources needed to accomplish a task given the initial capabilities of a set of agents sorted in maximal task coverage order (Fig. 2). In addition, a small

Algorithm 1 Basic agent loop for cooperative distributed systems

```
set initial role to explore
active ← true
rounds ← 0
while (no termination condition) do
if (active) then
  sense environment
  act according to role
  broadcast information to neighbors
  active ← false
else
  read neighbors' new information, if any
  deliberate and select role
  active ← true
endif
rounds++
end while
```

penalty is given to capabilities not relevant to the task. The preference for a task is proportional to its coverage and the preference of the agents selected for the task, ensuring commitment to mostly completed tasks.

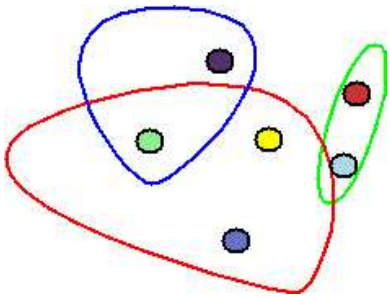


Fig. 2. Greedy Set Cover of a task decomposed into 6 subtasks (dots) with 3 agents and one overlapping capability.

4 Semi-Centralized Coordination Algorithms

Semi-centralized algorithms were found to be both practical and efficient in the large-scale coordination of agents [5] and lend themselves well to the widely used contract net protocol. The level of specificity in the planning of large groups does not extend to individual behaviors. DM&R planning in the National Response Plan [6] provides specific guidelines at the lowest geographical and organizational level but leaves room for self-organization. Semi-centralization through team

formation enables the continuous control of decentralized decision-making to achieve planned objectives and maintain situation awareness through data fusion at the global level. We first describe a cycle-based self-organizing algorithm for the formation of "cluster heads" and then our extension of this algorithm to take into account environmental demands.

4.1 Low-Energy Adaptive Clustering Hierarchy (LEACH) Algorithm

The LEACH algorithm [7] is a stochastic adaptive algorithm for energy-efficient communication in wireless sensor networks in the task allocation of aggregating and routing messages back to the base station (BS). Because of the limitation on battery power, the task should be fairly distributed among the nodes. In addition, aggregating the data to reduce noise before sending it to the BS is more efficient. Rotating this "cluster head" role among the nodes will minimize the overall energy consumed and allow the battery power to get replenished through solar energy. A round in the algorithm includes a setup phase establishing a transmission schedule to maximize bandwidth utilization and a steady-state phase where data fusion occurs and the aggregated messages are actually transmitted. It is assumed that the percentage of nodes that should take up this role is known a priori by the agents. An agent i assumes the role of "cluster head" if the stochastic probability is below a threshold T, determined as follows:

$$T(i) = \begin{cases} \frac{P}{1-P*(r \bmod \frac{1}{P})} & if\, i \in G \\ 0 & otherwise \end{cases} \tag{2}$$

where P is the desired percentage of cluster heads known a priori, r is the current round, and G is the set of agents that have not been cluster heads for the past $\frac{1}{P}$ rounds. If below threshold, the agent will advertise its services. Otherwise, the agents elect as their leader the closest agent according to the advertisements received.

4.2 Extension of the LEACH Algorithm

The LEACH algorithm assumes that (1) the activation percentage is given a priori and (2) the activation duration during which a schedule is propagated and messages are transmitted to the base station is fixed. This works well for sensor networks (e.g. unmanned aerial vehicles) where the number of nodes is known in advance and the only mission is to report back to the BS. This algorithm needs to be adapted to act as a relay in a mobile ad hoc network (i.e, transmit messages from any node to any other nodes) and to autonomously adjust to the number of nodes in the network. The time interval allocated to be a "cluster head" need not be limited to a single transmission to the BS and has to adapt to the needs of the network.

Adaptive Team Formation (ATF) algorithm If an agent does not assume a network role or "cluster head," it will receive only advertisement messages, and will send only election messages. As long as it receives advertisement messages, it does not have to compete for the network role. However, if everybody assume the same strategy, no service will be provided. The key idea is to predict the correct individual phase to alternate between roles based not only on internal disposition but also on the state of the environment. A coverage metric as the number of agents reached over the total number of agents looking for the service measures the performance of this algorithm. The time-to-live (TTL) parameter, latency and communication range affect the propagation of messages and the coverage of a node.

In contrast to other adaptation problems where convergence of an agent to a fixed behavior (or role) is desired, congestion problems requires learning when to change behavior to resolve conflicts. Response thresholds in swarm intelligence [8] induce a dynamic task allocation depending (1) on the disposition of the agents and (2) the environmental demands. A simple reinforcement learning scheme allocates agents to the task by either raising or lowering their response threshold. In our problem, an increase in connectivity (due to proximity or communication range) should sensitize an agent to be a team leader but a decrease in advertisement messages should also be an incentive to assume the role. The stimulus s_i for an agent i to become a team leader at time t depends on the connectivity of the agents (i.e. the number of other agents within one hop) or degree d_i of the network node, the change in connectivity δ_i, and repulsion factor $\alpha \in (0, 1)$ as follows:

$$s_{i_0} = \frac{d_{i_0}}{d_{i_0} + 1} \tag{3}$$

$$s_{i_{t+1}} = s_{i_t} - \alpha \frac{\#Advertisements}{\#Elections + \#Advertisements + \epsilon} + \delta_{i_{t+1}} \tag{4}$$

Here $\epsilon > 0$ is a small constant that prevents division by 0. The agent's response threshold T_i at time t taking into account its internal disposition θ_i and external demands is then as follows:

$$T_{i_t} = \frac{\theta_{i_t}}{1 + e^{-s_{i_t}}} \tag{5}$$

The initial disposition $\theta_{i_o} \in (0, 1)$ of an agent can be a function of its battery power or other hardware capabilities. To avoid specialization and redistribute the manager task fairly among the agents according to their capabilities, θ_t is adjusted based on the "fatigue" of performing the task or the "boredom" of not performing the task measured in cycles as in the LEACH algorithm above (see Subsect. 4.1).

$$\theta_{i_t} = \theta_{i_0} * (r \, mod \, \frac{1}{\theta_{i_0}}) \tag{6}$$

where r is the number of elapsed rounds.

It is assumed that the agents can perform their deliberative task in selecting a role in one round and that roles are noncommittal. The leader determines a proper role allocation (Sect. 3) of the team by iterating through each task. Roles are allocated to the best ranking team based on coverage of the task and preferences. The process repeats again on the remaining agents and tasks until no team can be formed. Redundancy against message loss occurs when roles are reallocated either by the same manager agent in the next round or another manager agent. Algorithm 2 describes the combined process.

5 Experimental Evaluation

The experiments were conducted with RePast [9], an agent-based simulation and modeling tool where agents act concurrently in a decentralized manner on a $n \times n$ grid. Its powerful scheduling mechanism was used to model the asynchronous behavior of the agents. Communication between agents was implemented by transmitting messages to agents in a Moore neighborhood of 7 cells, eliminating cycles, and time-to-live parameter set to 6 hops. In addition, a 5% message loss proportional to distance was simulated.

Figure 3 compares the static coverage rates of the LEACH and ATF clustering algorithms without task allocation for a varying number of agents in fixed random locations on a 100×100 grid. Only cluster nodes relay messages to other agents. The agents were randomly initialized with a disposition rate varying in the [0,0.1] range. The swarm-based ATF algorithm provides a significantly better coverage albeit with a larger clustering rate for each node. Nodes were cluster nodes at a rate of ~0.5% in the LEACH algorithm while their rate was evaluated at ~0.8% in the ATF algorithm.

5.1 Coordination Metric

Because coordination is an emergent property of interactive systems, it can only be measured indirectly through the performance of the agents in accomplishing a task where a task is decomposed in subgoals. The more complex the task, the higher the number of subgoals needed to be achieved. While performance is ultimately defined in domain-dependent terms, there are some common characteristics. Performance in a task can be measured either as the number of steps taken to reach the goal, i.e. its time complexity, or as the amount of resources required, i.e. its space complexity. An alternative evaluation for coordination is

Algorithm 2 Adaptive Team Formation

```
active ←true
set repulsion rate
rounds ← 0
while (no termination condition) do
if (active) then
  read role allocation mesg
  perform step(s) in role
  generate a random number r
  update stimulus and disposition
  T←estimate threshold
  if (r < T) then
    leader←true
    broadcast advertisement mesg
  else
    leader←false
    elect leader
    send election mesg to leader
  endif
  send role preferences mesgs to leader
  propagate all messages
  active←false
else
  read role preferences mesgs
  read role allocation mesgs
  read election mesgs
  read advertisement mesgs
  if (leader) then
    optimize task allocation for team
    send role allocation mesgs
  endif
  active←true
endif
rounds++
end while
```

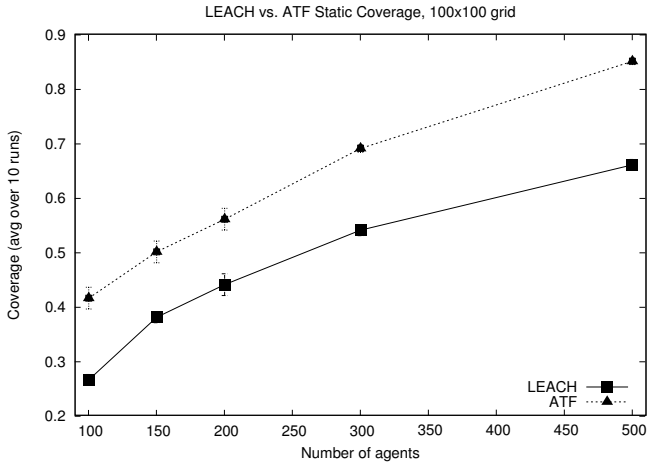


Fig. 3. LEACH vs. ATF static coverage comparison over 100 cycles

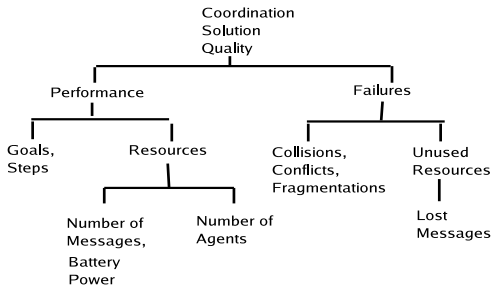


Fig. 4. Taxonomy of coordination solution quality

the absence of "failures", for example negative interactions such as collisions or lost messages. Figure 4 illustrates the taxonomy of coordination solution quality in pursuit games.

A combined coordination quality measure is defined as the harmonic mean of goals achieved g, resources expanded r and collisions c as follows :

$$g = \frac{\#Goals\,Achieved}{\#Goals} \tag{7}$$

$$r = \frac{\#agents}{log_2(\#Messages\,Received + 1) + \#agents} \tag{8}$$

$$c = \frac{\#agents}{log_2(\#Collisions + 1) + \#agents} \tag{9}$$

$$coordination = \frac{3grc}{gr + rc + cg} \tag{10}$$

Such a metric combining the different aspects of coordination can evaluate the tradeoff of performance and consuming bandwidth in large-scale tasks. The logarithms help normalize the distribution of the data across runs.

5.2 Fire/Rescue Problem

In our scenario, buildings are randomly created on a $n \times n$ grid with a random probability of being on fire and of spreading fire to adjacent buildings if not extinguished in time. Each fire or incident creates an emergency situation requiring up to k types of resources. In turn, each responder agent can provide up to k matching types of capabilities. There are a total of n capabilities and needs for each agent and incident ($n < k$). The problem consists of dynamically matching capabilities and needs with a team of agents. When a team of agents with the desired capabilities is situated near the incident within the agent's perception range p, the emergency will be removed. There are no scheduling constraints in matching resources but the overall resource requirements might increase over time as the fire spreads. Each agent has a perception range p and a typically greater communication range h to communicate with its neighbors. There are 4 types of messages in this scenario. Advertisement messages are broadcast while election messages are point-to-point. Received messages not matching the destination host are ignored using flooding, while ATF retransmits the message to the next leader node. Role preferences are communicated among agents (point-to-point to the leader with ATF and broadcast otherwise) that include the known targets and the associated preferences for covering each resource needed. The preferences are based on the distance to the incident and reflect the expected utility of the agent's capabilities. When observing an incident, a "resource needed" message is propagated among the agents describing the incident.

Figure 5 shows the coordination performance (Eq. 10) in this scenario with ATF where the elected leader node performs the network role of relaying messages. In the semi-centralized case, the leader node performs the managerial task of role allocation. In addition, one scout agent transmits additional observations to the leader nodes with retransmission to clients. In the distributed case, the role allocation task is performed implicitly by the agents. Both cases use the Greedy Set Cover algorithm for role allocation. Results show that self-organization and semi-centralization of role allocation incurs an overhead with a large number of agents and depends on other information available to the leader node for its performance with a smaller number of agents. There is a significance difference under ATF with 50 agents (t-test p-value = 0.002) between fully distributed role allocation and semi-centralized role allocation.

6 Related Work

Workpad [10] has proposed a 2-layer P2P architecture where the static internet backend provides the information services necessary to first responders in

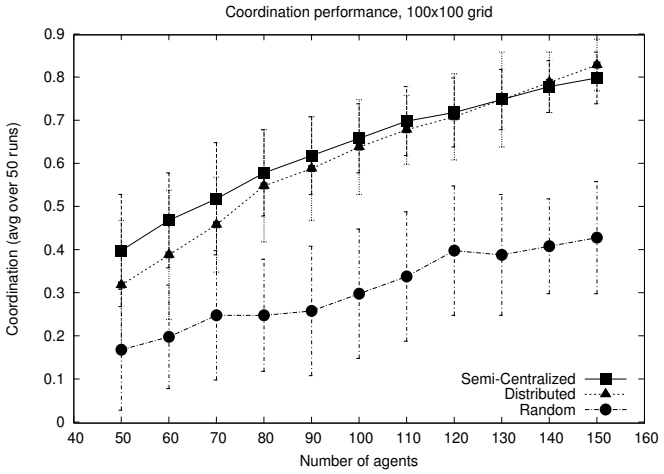


Fig. 5. Comparative coordination performance with ATF ($\alpha = 0.05$) for 20 incidents and a varying number of agents.

a tethered mobile ad hoc network. The scenarios explored an architecture for a coordination layer on top of the network layer where a team leader would reallocate tasks to solve predicted fragmentation of the network due to the mobility of the agents. In this paper we explored in detail the algorithms for role allocation and for selecting a team leader in a self-organized way.

Cooperative mediation [11] combines distributed role allocation with partial centralization. An agent, acting as mediator, recommends value changes (role assignments) to neighboring agents to optimize local subproblems. If a current solution is different from an optimal solution, the mediator transmits repairs to the relevant agents. Agents are prioritized to act as mediator based on the size of their "social knowledge." If a solution cannot be found, the neighboring agents transmit their constraints which could involve other agents enlarging the context of the subproblem. Cooperative mediation achieves a global optimal solution in a distributed way by exploiting the substructure of the problem. If no local optimal solution can be found, the mediator will progressively enlarge its context until an optimal global solution is found. Similarly, the ATF approach uses the degree of connectivity as a stimulus to influence the tendency of an agent to be a team leader but the election of a leader is explicit. A team leader divides the search space according to the substructure of the problem but does not attempt to reach a more global solution in this paper. The role of the team leader in ATF is not only to coordinate other agents in solving a task but also to coordinate the information sharing between agents.

7 Conclusion

We have presented applicable coordination algorithms and introduced a tiered P2P architecture to unify the different communication and coordination dimensions of DM&R with possible applications to other complex environments such as battlespace management. In addition, a novel self-organized and semi-centralized algorithm, ATF, has been introduced extending the LEACH algorithm to adaptive team formation. Semi-centralization is important to achieve planned objectives with bounded resources and to integrate disparate systems. Experimental evaluations of role allocation algorithms for heterogeneous agents have been presented in the fire/rescue domain along with a coordination metric that takes into account communication costs as well as partial goals achieved. Future work will include a more complex scenario where the leader nodes communicate between themselves to reach a more global solution.

References

1. Scerri, P., Pynadath, D., Schurr, N., Farinelli, A., Gandhe, S., Tambe, M.: Team oriented programming and proxy agents: The next generation. Workshop on Programming MultiAgent Systems, AAMAS 2003 (2003)
2. Project JXTA. http://www.jxta.org last accessed, 02/01/2007.
3. Abramson, M.: Three myths about roles. Technical report, American Association of Artificial Intelligence, Arlington, VA (2005) Fall Symposium Workshop on Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems.
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 2nd edn. MIT Press (2001)
5. Scerri, P., Vincent, R., Mailler, R.: Comparing three approaches to large-scale coordination. In: Proceedings of the Large-Scale Coordination Workshop at AAMAS. (2004)
6. Department of Homeland Security: National Response Plan. http://dhs.gov/xprepresp/publications last accessed, 01/23/2007.
7. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii IEEE International Conference on System Sciences. (2000)
8. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: from Natural to Artificial Systems. Oxford University Press (1999)
9. North, M.J., Collier, N.T., Vos, J.R.: Experiences creating three implementations of the repast agent modeling toolkit. ACM Transactions on Modeling and Computer Simulation **16**(1) (January 2006) 1–25
10. Mecella, M., Catarci, T., Angelaccio, M., Buttarazzi, B., Krek, A., Dustdar, S., Vetere, G.: Workpad: an adaptive peer-to-peer software infrastructure for supporting collaborative work of human operators in emergency/disaster scenarios. In: IEEE International Symposium on Collaborative Technologies and Systems (CTS 2006). (2006)
11. Mailler, R., Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation. In: International Conference on Autonomous Agents and Multiagent Systems. (2004)

Large-scale Organizational Computing requires Unstratified Paraconsistency and Reflection

Carl Hewitt

MIT EECS (Emeritus)
at alum.mit.ed try carlhewitt

Abstract. *Organizational Computing* is a computational model for using the principles, practices, and methods of human organizations. Organizations of Restricted Generality (*ORGs*) have been proposed as a foundation for Organizational Computing. ORGs are the natural extension of Web Services, which are rapidly becoming the overwhelming standard for distributed computing and application interoperability in Organizational Computing. The thesis of this paper is that large-scale Organizational Computing requires paraconsistency and reflection for organizational practices, policies, and norms.

Paraconsistency is required because the practices, policies, and norms of large-scale Organizational Computing are pervasively inconsistent. By the standard rules of logic, anything and everything can be inferred from an inconsistency, *e.g.,* "*The moon is made of green cheese*." The purpose of paraconsistent logic is to develop principles of reasoning so that such irrelevances cannot be inferred while preserving all natural inferences that do not explode in the face of inconsistency.

Reflection is required in order that the practices, policies, and norms can mutually refer to each other and make inferences.

Keywords: Co-ordination, Concurrency, Direct Logic, Inconsistency, Institutions, Mental Agents, Norms, Organizational Computing, ORGs (Organizations of Restricted Generality), Norms, Paraconsistency, Policies, Practices, Reflection.

Introduction

Organizational Computing is the metaphor of using an organizational model for computation; *i.e.,* computers using the principles, methods and practices of human organizations. Organizations of Restricted Generality (ORGs) have been proposed as a foundation for Organizational Computing [Hewitt and Inman 1991]. *ORGs are the natural extension of Web Services, which are rapidly becoming the overwhelming standard for distributed computing and application interoperability in Organizational Computing.* Microsoft, IBM, Oracle, SAP, and just about every Fortune 500 company are betting on Web Services.

The plan of this paper is as follows:

1. Introduce Organizational Computing and ORGs (Organizations of Restricted Generality) and describe the principles and practices by which they operate.
2. Develop the thesis that inconsistency is the norm for large-scale Organizational Computing.

3. Explain the limitations of classical logical reasoning for inconsistent information.
4. Introduce a system of Direct Logic that provides inference capabilities needed for large-scale Organizational Computing

Organizational Computing

Organizational Computing is a computational model for using principles, practices, and methods of human organizations. Organizations of Restricted Generality (*ORGs*) have been proposed as a foundation for Organizational Computing. In general

- ORGs mirror the structure of large-scale human organizations.
- ORGs are a natural extension Web Services, which are the standard for distributed computing and software application interoperability in large-scale Organizational Computing.
- ORGs are structured by *Organizational Commitment* [Jennings 1993; Noriega 1997; Singh and Huhns 2005], which is a special case of *Physical Commitment* [Hewitt 2006b] that is defined to be *information pledged*.[1]
- In many cases, humans will take part in the operation of an ORG. For example, in a credit card ORG, a particular credit decision may be reviewed by a human before being decided.

Inconsistency is the Norm in Large-scale Organizational Computing

The development of Organizational Computing and the extreme dependence of our society on these systems have introduced new phenomena. These systems have pervasive inconsistencies among and within the following:

- *Norms* that express how systems can be used and tested in practice
- *Policies* that expresses over-arching justification for systems and their technologies
- *Practices* that expresses implementations of systems

Different parties are responsible for constructing, evolving, justifying and maintaining practices, norms, and operations for large-scale Organizational Computing. In specific cases any one consideration can trump the others. Sometimes debates over inconsistencies can become quite heated, *e.g.,* between sales, engineering and finance.

Furthermore there is no evident way to divide up practices, norms, and operations into meaningful, consistent microtheories for large-scale Organizational Computing. *Organizations such as Microsoft, the US government, and IBM have tens of thousands of employees pouring over large Organizational Computing systems with hundreds of millions of lines of documentation, code, and use cases attempting to cope* [Rosenberg 2007].

[1] For example organizational authority can be manifested in physical commitment. Possession of a key enabling decryption of a message manifests authority to read the message. Also, for example, there is a Microsoft ORG that has the power to sign code that will subsequently be accepted as authoritative by hundreds of millions of computers.

Adapting a metaphor that Karl Popper [1962] used for science, the bold structure of a large Organizational Computing system rises, as it were, above a swamp. It is like a building erected on piles. The piles are driven down from above into the swamp, but not down to any natural or given base; and when we cease our attempts to drive our piles into a deeper layer, it is not because we have reached bedrock. We simply pause when we are satisfied that they are firm enough to carry the structure, at least for the time being. Or perhaps we do something else more pressing. Under some piles there is no rock. Also some rock does not hold.

Different parties (management, engineering, marketing, sales, *etc*.) are responsible for constructing, evolving, justifying and maintaining documentation, use cases, and code for large, human-interaction, Organizational Computing systems. In specific cases any one consideration can trump the others. Sometimes debates over inconsistencies among the parts can become quite heated, *e.g.,* among engineering, marketing, and sales. ***In large Organizational Computing systems, policies, practices, and norms all co-evolve to eliminate old inconsistencies and produce systems with new inconsistencies. However, no one knows what they are or where they are located!***

Furthermore there is no evident way to divide up the code, documentation, and use cases into meaningful, consistent microtheories for human-computer interaction. *Organizations such as Microsoft, the US government, and IBM have tens of thousands of employees pouring over hundreds of millions of lines of documentation, code, and use cases attempting to cope with their Organizational Computing Systems. In the course of time almost all of this code will interoperate using Web Services. A large Organizational Computing system is never done* [Rosenberg 2007]**.**

The thinking in almost all scientific and engineering work has been that models (also called theories or microtheories) should be internally consistent, although they could be inconsistent with each other.

Limitations of Classical Logic for Inconsistent Information

As mentioned above, a limitation of classical logic for inconsistent theories is that it supports the principle that from an inconsistency anything can be inferred, *e.g. "The moon is made of green cheese."* This principle will be called IGOR for Inconsistency in Garbage Out Redux. IGOR can be formalized as follows:[2] $\Phi, \neg\Phi \vdash \Psi$

The IGOR principle of classical logic may not seem very intuitive! So why is it included in classical logic?

The IGOR principle is readily derived from the following principles of classical logic:

- *Full indirect inference*: $(\Psi \vdash \Phi, \neg\Phi) \rightarrow (\vdash \neg\Psi)$ which can be justified in classical logic on the grounds that if Ψ infers a contradiction in a consistent theory then Ψ must be false. In an inconsistent theory, full indirect inference leads to explosion by the following derivation in classical logic:

  $$\Phi,\neg\Phi \vdash (\neg\Psi \vdash \Phi,\neg\Phi) \vdash (\neg\neg\Psi) \vdash \Psi$$

[2] Using the symbol $\vdash_T$ to mean "infers in a theory T" and $\rightarrow$ to mean mathematical logical implication.

- *Disjunction introduction*: **($\Psi \vdash (\Psi \vee \Phi)$)** which in classical logic would say that if **Ψ** is true then **($\Psi \vee \Phi$)** is true regardless of whether **Φ** is true. In an inconsistent theory, disjunction introduction leads to explosion via the following derivation in classical logic:

  $\Phi, \neg\Phi \vdash (\Phi \vee \Psi), \neg\Phi \vdash \Psi$

Direct Logic

Direct Logic[3] is a powerful inference system for large-scale Organizational Computing with the following goals [Hewitt 2006a 2007a]:[4]

- Provide a paraconsistent unstratified reflective mathematical foundation for inference and reflection in large-scale Organizational Computing. Unstratified inference and reflection means that Direct Logic is its own metatheory. Paraconsistent means that a single inconsistency in a theory does not infer everything.
- Formalize a notion of "direct" inference for paraconsistent theories.
- Support all "natural" deductive inference in paraconsistent theories that does not blow up in the face of an inconsistency.
- Provide increased safety in reasoning about large-scale Organizational Computing using paraconsistent theories.

Multiple ORGs can make use of Direct Logic is a distributed decentralized fashion using a network of multiple paraconsistent theories. There is no requirement for an ORG to maintain a unified coherent mental state (as in Mental Agents [Hewitt 2007b]).

Reification in Organizational Computing

Every sentence Ψ has reification that is given by $\ulcorner\Psi\urcorner \in \text{Sentences} \subseteq \text{XML}$. Similarly every $\sigma \in \text{Sentences}$ has an anti-reification that is the sentence given by $\llcorner\sigma\lrcorner$. The following holds

Reification and anti-reification are needed for large Organizational Computing systems so that that practices, policies, and norms can mutually speak about what has been said and its meaning.

The practices, policies, and norms are becoming increasingly *mutually reflective* in that they refer to and make use of each other. *E.g.,*

- Practices can be inferred by specialization of policies and can be dynamically checked against policies. Also practices can be dynamically searched for and invoked on the basis of policies.
- Policies can be checked against each other and against practices using model checking.

[3] Direct Logic is distinct from the Direct Predicate Calculus [Ketonen and Weyhrauch 1984].
[4] How these goals are realized is described in the appeendix to this paper.

- Norms can be generated by inference from policies and proposed by generalization from practices.

Principle. ***Reflection preserves equivalence.***
$(\Psi \longleftrightarrow \Phi) \rightarrow (\lfloor\lceil\Psi\rceil\rfloor \longleftrightarrow \lfloor\lceil\Phi\rceil\rfloor)$

The big issue for reification is as follows:[5]
When is it the case that $\lfloor\lceil\Psi\rceil\rfloor$ ***is equivalent to*** Ψ**?**

Fixed Point Theorem

Theorem [after Carnap 1934 pg 91][6]:
Let $f \in (\textbf{Sentences} \rightarrow \textbf{Sentences})$

$\vdash (\lfloor \textbf{Fix}(f) \rfloor \longleftrightarrow \lfloor f\ (\textbf{Fix}(f)) \rfloor)$
where $\textbf{Fix}(f) \equiv \Theta\ (\Theta)$
where $\Theta \equiv \lambda(P)\ f\ (\ P\ (P))$

Proof
$\lfloor \textbf{Fix}(f) \rfloor \longleftrightarrow \lfloor \Theta(\Theta) \rfloor$
$\longleftrightarrow \lfloor \lambda(P)\ f\ (\ P(P))\ (\Theta) \rfloor$
$\longleftrightarrow \lfloor f\ (\ \Theta(\Theta)) \rfloor$
$\longleftrightarrow \lfloor f\ (\textbf{Fix}(f)) \rfloor$

The Liar Paradox

The Liar Paradox goes back at least as far as the Greek philosopher Eubulides of Miletus who lived in the fourth century BC. It could be put as follows:

LiarStatement is defined to be: "The negation of **LiarStatement** holds."

From its definition, **LiarStatement** holds if and only if it doesn't!

The argument can be formalized using the fixed point theorem and the diagonal argument [Cantor 1890, Zermelo 1908] in the following way:

$\textbf{LiarStatement} \equiv \lfloor \textbf{Fix}(\text{Diagonal}) \rfloor$
where $\text{Diagonal} \equiv \lambda(\sigma) \lceil \neg \lfloor \sigma \rfloor \rceil$

The Liar Paradox can be stated as follows:[7] $\textbf{LiarStatement} \longleftrightarrow \neg\ \textbf{LiarStatement}$

Argument for the Liar Paradox[8]
$\textbf{LiarStatement} \longleftrightarrow \lfloor \textbf{Fix}(\text{Diagonal}) \rfloor$
$\longleftrightarrow \lfloor \text{Diagonal}\ (\textbf{Fix}(\text{Diagonal})) \rfloor$ *; by the fixed point theorem*
$\longleftrightarrow \lfloor \lambda(\sigma) \lceil \neg \lfloor \sigma \rfloor \rceil\ (\textbf{Fix}(\text{Diagonal})) \rfloor$
$\longleftrightarrow \lfloor \lceil \neg \lfloor \textbf{Fix}(\text{Diagonal}) \rfloor \rceil \rfloor$
$\longleftrightarrow \lfloor \lceil \neg\ \textbf{LiarStatement} \rceil \rfloor$
$\longleftrightarrow \neg\ \textbf{LiarStatement}$ *; this step is* ***not*** *valid in Direct Logic*

[5] As explained in the sections, below reflection can introduce spurious inconsistencies unless it is handled carefully.
[6] Credited in Kurt Gödel, *Collected Works* vol. I, p. 363, ftn. 23.
[7] As explained below, the Liar Paradox does not hold in Direct Logic.
[8] As explained below, this argument is *not* valid in Direct Logic.

Disadvantages of stratified reflection

To avoid inconsistencies in mathematics (e.g., Liar Paradox, Russell's Paradox, Curry's Paradox, *etc*.), some restrictions are needed around self-reference. The question is how to do it [Feferman 1984, Restall 2006].

The approach which is currently standard in mathematics is the Tarskian framework of stratifying theories into a hierarchy of metatheories in which the semantics of each theory is formalized in its metatheory [Tarski 1933].

According to Feferman [1984]:

> *"…natural language abounds with directly or indirectly self-referential yet apparently harmless expressions—all of which are excluded from the Tarskian framework."*

Large Organizational Computing systems likewise abound with directly or indirectly self-referential statements in reasoning about their use cases, documentation, and code that are excluded by the Tarskian framework. Consequently the Tarskian framework is not very suitable for Organizational Computing.

Logical Reflection Principle for Organizational Computing

The ***Logical Reflection Principle*** for Direct Logic is that for each $\sigma \in \mathbf{Sentences}$:

$$\vdash (\sigma \in \mathbf{Admissibles}_{\mathsf{T}} \rightarrow (\lfloor \ulcorner \lfloor \sigma \rfloor \urcorner \rfloor \longleftrightarrow_{\mathsf{T}} \lfloor \sigma \rfloor))$$

Of course, the above criterion begs the questions of which sentences are Admissible in T! A proposed answer is provided by the following:

Criterion of Admissibility: A sentence is Admissible for T ***if and only if***

$$(\neg\Psi) \rightarrow_{\mathsf{T}} \vdash_{\mathsf{T}} \neg\Psi$$

The ultimate suitability of the Admissibility Criterion for large Organizational Computing systems remains to be determined.

The argument of the Liar Paradox is not valid for paraconsistent theories in Direct Logic.

The argument of the Liar Paradox is not valid in Direct Logic because:

$$\vdash \ulcorner \neg\, \mathbf{LiarStatement} \urcorner \notin \mathbf{Admissibles}$$

and consequently the Logical Reflection Principle of Direct Logic does not apply to $\ulcorner \neg\, \mathbf{LiarStatement} \urcorner$.[9]

Conclusion

This paper introduces Organizational Computing and ORGs (Organizations of Restricted Generality) and describes the principles and practices by which they operate. It develops the thesis that inconsistency is the norm for large-scale Organizational Computing. The limitations of classical logical reasoning for inconsistent information are explained. A powerful inference system called Direct

[9] Likewise Russell's Paradox and Curry's Paradox are not valid for paraconsistent theories in Direct Logic.

Logic is introduced that provides inference capabilities needed for large-scale Organizational Computing including unstratified paraconsistency and reflection.

Acknowledgments

Sol Feferman, Mike Genesereth, David Israel, Bill Jarrold, Ben Kuipers, Pat Langley, Vladimir Lifschitz, Frank McCabe, John McCarthy, Fanya S. Montalvo, Peter Neumann, Ray Perrault, Mark Stickel, Richard Waldinger, and others provided valuable feedback at seminars at Stanford, SRI, and UT Austin to an earlier version of the material in this paper. For the AAAI Spring Symposium'06, Ed Feigenbaum, Mehmet Göker, David Lavery, Doug Lenat, Dan Shapiro, and others provided valuable feedback. At MIT Henry Lieberman, Ted Selker, Gerry Sussman and the members of Common Sense Research Group made valuable comments. Reviewers for AAMAS '06 and '07, KR'06, COIN@AAMAS'06 and IJCAR'06 made suggestions for improvement.

In the logic community, Mike Dunn, Sol Feferman, Mike Genesereth, Tim Hinrichs, Mike Kassoff, John McCarthy, Chris Mortensen, Graham Priest, Dana Scott, Richard Weyhrauch and Ed Zalta provided valuable feedback. Dana Scott made helpful suggestions on reflection and incompleteness. Richard Waldinger provided extensive suggestions that resulted in better focusing a previous version of this paper and increasing its readability. Sol Feferman reminded me of the connection between Admissibility and Π_1. Discussion with Pat Hayes and Bob Kowalski provided insight into the early history of Prolog.

Communications from John McCarthy and Marvin Minsky suggested making common sense a focus. Mike Dunn collaborated on looking at the relationship of Propositional Direct Logic to R-Mingle. Greg Restall pointed out that R-Mingle differs from the propositional fragment of Direct Logic because R-Mingle is a Relevance Logic. Gerry Allwein and Jeremy Forth made detailed comments and suggestions for improvement. Bob Kowalski and Erik Sandewall provided helpful pointers and discussion of the relationship with their work. Discussions with Ian Mason and Tim Hinrichs helped me develop Löb's theorem for paraconsistent theories. Fanya S. Montalvo provided valuable comments.

References

Gul Agha, Ian Mason, Scott Smith, and Carolyn Talcott. "A foundation for Actor computation." *Journal of Functional Programming.* 1997.

Geof Bowker, Susan L. Star, W. Turner, and Les Gasser, (Eds.) *Social Science Research, Technical Systems and Cooperative Work*. Lawrence Earlbaum. 1997.

Rudolph Carnap. Logische Syntax der Sprache. (*The Logical Syntax* of Language Open Court Publishing 2003) 1934.

Ole-Johan Dahl and Kristen Nygaard. "Class and subclass declarations" *IFIP TC2*. May 1967.

Solomon Feferman. "Toward Useful Type-Free Theories, I" in *Recent Essays on Truth and the Liar Paradox.* Ed. Robert Martin. Claraendon Press. 1984.

Kurt Gödel. "On formally undecidable propositions of *Principia Mathematica"* translated by Bernard Meltzer. Basic Books. 1962. Monatshefte für Mathematik und Physik, Vol. 38, pp. 173-198. 1931. http://home.ddc.net/ygg/etext/godel/

Carl Hewitt, Peter Bishop and Richard Steiger. "A Universal Modular Actor Formalism for Artificial Intelligence" *IJCAI* 1973.

Carl Hewitt and Jeff Inman. "DAI Betwixt and Between: From 'Intelligent Agents' to Open Systems Science" *IEEE Transactions on Systems, Man, and Cybernetics.* Nov. /Dec. 1991.

Carl Hewitt 2006a. "The repeated demise of logic programming and why it will be reincarnated" *What Went Wrong and Why.* Technical Report SS-06-08. March 2006.

Carl Hewitt. 2006b. "What is Commitment? Physical, Organizational, and Social" *COIN@AAMAS'06*. (Revised version in *Coordination, Organizations, Institutions and Norms in Multi-Agent Systems II*. Edited by Javier Vázquez-Salceda and Pablo Noriega. Springer Verlag. 2007) April 2006.

Carl Hewitt 2007a. "Common sense for paraconsistency and concurrency using unstratified inference and reflection." Submitted for publication to AI Journal special issue on common sense". 2007.

Carl Hewitt 2007b. "The Downfall of Mental Agents in the Implementation of Large Software Systems" AAAI Magazine issue on *What went wrong and why.* 2007.

Nicholas Jennings. "Commitments and conventions: The foundation of coordination in multi-agent systems" *Knowledge Engineering Review.* 3. 1993.

Jussi Ketonen and Richard Weyhrauch. "A decidable fragment of Predicate Calculus" *Theoretical Computer Science.* 1984.

Bill Kornfeld and Carl Hewitt. "The Scientific Community Metaphor" *IEEE Transactions on Systems, Man, and Cybernetics.* January 1981.

Robert Kowalski. "The Logical Way to be Artificially Intelligent." *CLIMA VI.* Springer Verlag. 2006

Martin Löb. "Solution of a problem of Leon Henkin.". *Journal of Symbolic Logic*. Vol. 20. 1955.

Thomas Malsch and Ingo Schulz-Schaeffer. "Socionics: Sociological Concepts for Social Systems of Artificial (and Human) Agents" *Journal of Artificial Societies and Social Simulation.* January, 2007.

Robin Milner. "Elements of interaction: Turing award lecture." *CACM.* January 1993.

Pablo Noriega. *Agent Mediated Auctions: The Fishmarket Metaphor*. Ph.D. Barcelona. 1997.

Graham Priest and Koji Tanaka. "Paraconsistent Logic" *The Stanford Encyclopedia of Philosophy.* Winter 2004.

Greg Restall. "Curry's Revenge: the costs of non-classical solutions to the paradoxes of self-reference" (to appear in *The Revenge of the Liar* ed. J.C. Beall. Oxford University Press. 2007) July 12, 2006.

Scott Rosenberg. *Dreaming in code.* Crown Publishing. 2007.

John Barkley Rosser. "Extensions of Some Theorems of Gödel and Church" *Journal of Symbolic. Logic.* 1(3) 1936.

Munindar Singh and Michael Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons. 2005.

Michael Wooldridge. *Reasoning about Rational Agents*. MIT Press. 2000.

Appendix. Principles of Direct Logic

This appendix discusses fundamental principles of Direct Logic focusing on inference in a theory T ($\vdash_{\mathsf{T}}$) where T may possibly be inconsistent. All of the principles below concerning $\vdash_{\mathsf{T}}$ apply equally well to the overreaching classical inference relationship for the classical fragment of Direct Logic ($\vdash$) Consequently, inference in Direct Logic is unstratified and Direct Logic is its own metatheory.

Direct Principles

Direct principles that connect sentences together by direct reasoning are as follows:

Reiteration: $\vdash (\Psi \vdash_{\mathsf{T}} \Psi)$ *; a sentence infers itself*

Exchange: $\vdash (\Psi, \Phi \vdash_{\mathsf{T}} \Phi, \Psi)$
; the order in which sentences are written does not matter

Residuation: $\vdash ((\Psi, \Phi \vdash_{\mathsf{T}} \Theta) \longleftrightarrow (\Psi \vdash_{\mathsf{T}} (\Phi \vdash_{\mathsf{T}} \Theta)))$
; hypotheses may be introduced and discharged

Monotonicity: $\vdash ((\Psi \vdash_{\mathsf{T}} \Phi) \rightarrow (\Psi, \Theta \vdash_{\mathsf{T}} \Phi))$
; an inference remains correct if a new hypothesis is added

Dropping: $\vdash ((\Psi \vdash_{\mathsf{T}} \Phi, \Theta) \rightarrow (\Psi \vdash_{\mathsf{T}} \Phi))$
; an inference remains correct if extra conclusions are dropped

Contraction: $\vdash ((\Psi \vdash_{\mathsf{T}} (\Psi \vdash_{\mathsf{T}} \Phi)) \longleftrightarrow (\Psi \vdash_{\mathsf{T}} \Phi))$
; an inference remains correct if the same hypothesis is dropped or added

Combination Principles

The following combination principles concern combining different inferences together:

Independent inference:

$\vdash ((\vdash_{\mathsf{T}} \Psi) \wedge (\vdash_{\mathsf{T}} \Phi)) \rightarrow (\vdash_{\mathsf{T}} \Psi, \Theta))$ *; inferences can be combined*

Transitivity:

$\vdash (((\Psi \vdash_{\mathsf{T}} \Phi) \wedge (\Phi \vdash_{\mathsf{T}} \Theta)) \rightarrow (\Psi \vdash_{\mathsf{T}} \Theta))$ *; inference is transitive*

Theory Principles

The paraconsistent theory principles are as follows:

Faithfulness: $\vdash ((\vdash_{\mathsf{T}} \vdash_{\mathsf{T}} \Psi) \rightarrow (\vdash_{\mathsf{T}} \Psi))$

Adequacy: $\vdash ((\vdash_{\mathsf{T}} \Psi) \rightarrow (\vdash_{\mathsf{T}} \vdash_{\mathsf{T}} \Psi))$

Detachment: $\vdash ((\vdash_{\mathsf{T}} \Psi, \Psi \vdash_{\mathsf{T}} \Phi) \rightarrow \vdash_{\mathsf{T}} \Phi)$

Soundness: $\vdash ((\Psi \vdash_{\mathsf{T}} \Phi) \rightarrow ((\vdash_{\mathsf{T}} \Psi) \rightarrow \vdash_{\mathsf{T}} \Phi))$

Inferences have proofs: $\vdash ((\Psi \vdash_{\mathsf{T}} \Phi) \rightarrow \vdash_{\mathsf{T}} (\Psi \vdash_{\mathsf{T}} \Phi))$

Logical Connectives $\wedge$, $\vee$, and $\rightarrow$

The logical connectives $\wedge_T$, $\vee_T$, and $\rightarrow_T$ are defined in terms of $\vdash_T$ and $\neg$ as follows[10].

Definition of $\wedge_T$

The "definition" of conjunction in theory T ($\wedge_T$) is

$$((\Psi \wedge_T \Phi) \vdash_T \Theta) \cong (\Psi,\Phi \vdash_T \Theta)$$

$$(\Phi \vdash_T (\Psi \wedge_T \Theta)) \cong (\Phi \vdash_T \Psi,\Theta)$$

Definition of $\vee_T$

The definition of disjunction in theory T ($\vee_T$) is

$$\Psi \vee_T \Phi \cong (((\neg\Psi) \vdash_T \Phi) \wedge ((\neg\Phi) \vdash_T \Psi)))$$

Note that the semantics of $\vee_T$ is inferential as opposed to truth functional.

Theorem. $(\vdash_T \Psi, \Phi) \rightarrow \vdash_T (\Psi \vee_T \Phi)$

The ***Principle of Disjunctive Cases*** is as follows:

$$\vdash (((\Psi \vee_T \Phi) \wedge (\Psi \vdash_T \Theta) \wedge (\Phi \vdash_T \Theta)) \rightarrow (\vdash_T \Theta))$$

Definition of $\rightarrow_T$

The definition of implication in theory T ($\rightarrow_T$) is

$$\Psi \rightarrow_T \Phi \cong (\Psi \vdash_T \Phi) \wedge (\neg\Phi \vdash_T \neg\Psi)$$

Note that the semantics of $\rightarrow_T$ is inferential as opposed to truth functional.

Theorem (Transitivity of $\rightarrow_T$): $\vdash (((\Psi \rightarrow_T \Phi) \wedge (\Phi \rightarrow_T \Theta)) \rightarrow (\Psi \rightarrow_T \Theta))$

Double Negation Elimination

The Principle of **Double Negation Elimination** is as follows:

$$\vdash \vdash_T (\neg\neg \Psi \cong \Psi)$$

Theorem: $\vdash \vdash_T (\Psi \vee_T \neg\Psi)$

Theorem: $\vdash \vdash_T ((\Psi \rightarrow_T \Phi) \cong (\neg\Psi \vee_T \Phi) \cong (\neg\Phi \rightarrow_T \neg\Psi))$

Theorem: **Residuation for Implication**

$$\vdash ((\Psi,\Phi \vdash_T \Theta) \longleftrightarrow (\Psi \rightarrow_T (\Phi \rightarrow_T \Theta)))$$

Equivalences for $\wedge_T$, $\vee_T$ ***and*** $\rightarrow_T$

The usual equivalences hold for conjunction and disjunction:

$(\Psi \wedge_T \Psi) \cong \Psi$
$(\Psi \wedge_T \Phi) \cong (\Phi \wedge_T \Psi)$
$(\Psi \wedge_T (\Phi \wedge_T \Theta)) \cong ((\Psi \wedge_T \Phi) \wedge_T \Theta)$
$(\Psi \wedge_T (\Phi \vee_T \Theta)) \cong ((\Psi \wedge_T \Phi) \vee_T (\Psi \wedge_T \Theta))$
$\neg (\Psi \wedge_T \Phi) \cong \neg\Psi \vee_T \neg \Phi$

$(\Psi \vee_T \Psi) \cong \Psi$
$(\Psi \vee_T \Phi) \cong (\Phi \vee_T \Psi)$
$(\Psi \vee_T (\Phi \vee_T \Theta)) \cong ((\Psi \vee_T \Phi) \vee_T \Theta)$
$(\Psi \vee_T (\Phi \wedge_T \Theta)) \cong ((\Psi \vee_T \Phi) \wedge_T (\Psi \vee_T \Theta))$
$\neg (\Psi \vee_T \Phi) \cong \neg\Psi \wedge_T \neg \Phi$

[10] In these definitions, $\cong$ is used as a symbol for metalinguistic equivalence.

$(\Psi \rightarrow_T \Phi) \cong (\neg\Psi \vee_T \Phi) \cong (\neg\Phi \rightarrow_T \neg\Psi)$
$((\Psi \wedge_T \Phi) \rightarrow_T \Theta) \cong (\Psi \rightarrow_T (\Phi \rightarrow_T \Theta))$
$((\Psi \rightarrow_T (\Phi \wedge_T \Theta)) \cong ((\Psi \rightarrow_T \Phi) \wedge_T ((\Psi \rightarrow_T \Theta))$

Indirect inference

A major motivation for maintaining the consistency of mathematics is to allow Full Indirect Inference for the mathematical fragment of Direct Logic

$$(\Psi \vdash \Phi, \neg \Phi) \rightarrow \vdash \neg\Psi$$

Full Indirect Inference can be very inconvenient to do without in mathematical argument. For example the proof of the incompleteness of paraconsistent theories in this paper makes use of Full Indirect Inference.

Direct Logic supports direct versions of indirect inference for paraconsistent theories as follows

- ***Direct Indirect Inference:***
  $\vdash((\Psi \vdash_T \neg\Psi) \rightarrow (\vdash_T \neg\Psi))$ which states that a sentence can be disproved by showing that it infers its own negation.
- ***Right Meta Direct Indirect Inference:***
  $\vdash((\Psi \vdash_T (\vdash_T \neg\Psi)) \rightarrow \vdash_T \neg\Psi)$ which states that a sentence can be disproved by showing that it infers a proof of its own negation.
- ***Left Meta Direct Indirect Inference:***
  $\vdash(((\vdash_T \Psi) \vdash_T \neg\Psi)) \rightarrow \neg \vdash_T \Psi)$ which states that provability of a sentence can be disproved by showing that its provability infers its own negation.

Nontriviality principles for paraconsistent theories

Direct Logic supports the following nontriviality[11] principles for paraconsistent theories:

- ***Direct Nontriviality***: $\vdash((\neg \Psi) \vdash_T \neg \vdash_T \Psi)$ which states that if the negation of a sentence holds, then it cannot be proved
- ***Meta Nontriviality***: $\vdash((\vdash_T \neg \Psi) \vdash_T \neg \vdash_T \Psi)$ which states that if the negation of sentence can be proved, then it cannot be proved.

Incompleteness of Paraconsistent Theories

Incompleteness of a theory T (denoted by $\mathbf{Incomplete}[T]$) is defined to mean that there is some sentence such that it cannot be proved and neither can its negation. It can be formally defined as follows:

$$\exists\sigma\in\mathbf{Sentences}\ ((\neg \vdash_T \lfloor\sigma\rfloor) \wedge (\neg \vdash_T \neg\lfloor\sigma\rfloor))$$

[11] By definition a theory T is *nontrivial* if and only if there is a formula Ψ such that $\neg \vdash_T \Psi$.

Paraconsistent Incompleteness Theorem

Theorem: A paraconsistent theory is incomplete, i.e.,

$\vdash \forall T \in \mathbf{Theories}\ \ \mathbf{Incomplete}[T]$

Furthermore

$\mathbf{Paradox}_T \equiv \lfloor \mathbf{Fix}(\text{Diagonal}) \rfloor$

where $\text{Diagonal} \equiv \lambda(\sigma) \lceil \neg \vdash_T \lfloor \sigma \rfloor \rceil$

It is sufficient to prove the following:

1. $\vdash \neg \vdash_T \mathbf{Paradox}_T$
2. $\vdash \neg \vdash_T \neg \mathbf{Paradox}_T$

Proof. See [Hewitt 2007a].

Nested Meta Self Implication for Paraconsistent Theories

Nested Meta Self Implied sentences are those for which it is provable that they are implied by their proof.

Definition. $\mathbf{NestedMetaSelfImplications}_T \equiv \{\sigma \mid \vdash_T ((\vdash_T \lfloor \sigma \rfloor) \rightarrow_T \lfloor \sigma \rfloor)\}$

Theorem (After Löb [1955])**:**

If Ψ ***is Admissible and Nested Meta Self Implied for*** T, ***then*** $\vdash_T \Psi$

Proof. See [Hewitt 2007a].

Lemma. $\vdash \lceil \mathbf{Paradox}_T \rceil \in \mathbf{Admissibles}_T$

Corollary. $\vdash \lceil \mathbf{Paradox}_T \rceil \notin \mathbf{NestedMetaSelfImplications}_T$, *i.e.*

$\vdash \neg \vdash_T ((\vdash_T \mathbf{Paradox}_T) \rightarrow_T \mathbf{Paradox}_T)$