Lecture Notes in Artificial Intelligence 5428

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Jomi Fred Hübner Eric Matson
Olivier Boissier Virginia Dignum (Eds.)

Coordination, Organizations Institutions and Norms in Agent Systems IV

COIN 2008 International Workshops
COIN@AAMAS 2008, Estoril, Portugal, May 12, 2008
COIN@AAAI 2008, Chicago, USA, July 14, 2008
Revised Selected Papers

Springer

Volume Editors

Jomi Fred Hübner
Olivier Boissier
ENS Mines Saint-Etienne
42023 Saint-Etienne Cedex 02, France
E-mail:{hubner;boissier}@emse.fr

Eric Matson
Purdue University
West Lafayette, IN 70907, USA
E-mail: ematson@purdue.edu

Virginia Dignum
Utrecht University
3508TB Utrecht, The Netherlands
E-mail: virginia@cs.uu.nl

Library of Congress Control Number: Applied for

CR Subject Classification (1998): I.2.11, I.2, D.2, F.3, D.1, C.2.4, D.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-00442-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00442-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12624836 06/3180 5 4 3 2 1 0

Preface

Multi-agent systems (MAS) are often understood as complex entities where a multitude of agents interact, usually with some intended individual or collective goals. Such a view usually assumes some form of organization, or set of norms or conventions that articulate or restrain interactions in order to make them more effective, certain, or predictable for participants. Engineering 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 especially in applications on service-oriented computing, grid computing and ambient intelligence. These applications 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 also societal and organization-centric views.

However, agent autonomy is often needed for concretely implementing social order, because autonomous agents can intelligently adapt the designed organization to particular cases and can face unpredicted events. From this perspective autonomy can also be a possible source of internal change in the designed organizational constructs. Differently, autonomous behavior can also originate forms of self-organization which emerge out of local interactions and are only partially externally programmed. In such situations the self-organized order and the externally designed organization can even be in conflict.

These interdisciplinary issues have to be studied considering the analysis of the social, legal, economic, and technological dimensions of multi-agent organizations. In this context, the COIN workshops provide a space for the convergence of concerns and developments of MAS researchers that have been involved with these issues from the complementary perspectives of *coordination*, *organizations*, *institutions*, and *norms*. This year, the COIN workshops were hosted during AAMAS 2008 (May 12, Estoril, Portugal) and AAAI 2008 (July 14, Chicago, USA).

The papers contained in this volume are the revised and extended versions of a selection of papers presented and discussed in these two workshops. They are organized in five parts. The first part (from coordination to organization) groups papers focused on the coordination of agents where the organization is a result of the overall coordination process. The second part (from organization to coordination) takes the other direction. Considering that the system has an organization, the question is how to coordinate the agents belonging to this organization. In particular, how reputation can be used inside organizations to help the agents to better coordinate themselves. While the third part (formalization of norms and institutions) contains papers focused on the formalization of

important concepts related to norms and institutions, the papers of the fourth part (design of norms and institutions) are more concerned with the design, including the verification, of such concepts. The fifth part contains papers that apply the concepts of the workshops in applications and simulations.

We would like to thank the COIN Steering Committee for presenting us with the opportunity to organize these workshops. We also want to express our gratitude to the Program Committee members and additional reviewers of both events, to the participants of the workshops, and more particularly to the authors for their original contributions and further revisions for this volume. We thank the organizers of the AAMAS and AAAI events for hosting and supporting the organization of the COIN workshops. Finally, we would also like to acknowledge the support from Springer, in the person of Alfred Hofmann, for the publication of the COIN workshops since the first edition in 2004.

December 2008

Jomi F. Hübner
Olivier Boissier
Eric Matson
Virginia Dignum

Organization

Organizing Committees

COIN@AAMAS 2008

Jomi Fred Hübner	University of Blumenau, Brazil
Olivier Boissier	ENS Mines Saint-Etienne, France

COIN@AAAI 2008

Eric Matson	Purdue University, USA
Virginia Dignum	Utrecht University, The Netherlands

COIN Steering Committee

Christian Lemaître	Universidad Autonoma Metropolitana, Mexico
Eric Matson	Purdue University, USA
Guido Boella	University of Turin, Italy
Jaime Simão Sichman	University of São Paulo, Brazil
Javier Vázquez-Salceda	Universitat Politecnica de Catalunya, Spain
Julian Padget	University of Bath, UK
Nicoletta Fornara	University of Lugano, Italy
Olivier Boissier	ENS Mines Saint-Etienne, France
Pablo Noriega	IIIA-CSIC, Spain
Sascha Ossowski	Universidad Rey Juan Carlos, Spain
Virginia Dignum	Utrecht University, The Netherlands
Wamberto Vasconcelos	University of Aberdeen, UK

Program Committees

COIN@AAMAS 2008

Alessandro Ricci	Università di Bologna, Italy
Alexander Artikis	NCSR Demokritos, Greece
Andrea Omicini	Università di Bologna, Italy
Antonio Carlos da Rocha Costa	UCPEL, Brazil
Carles Sierra	IIIA-CSIC, Spain
Catherine Tessier	ONERA, France
Catholijn Jonker	Delft University, of Technology, The Netherlands
Christian Lemaître	Universidad Autónoma Metropolitana, Mexico
Christophe Sibertin-Blanc	IRIT, France
Cristiano Castelfranchi	ISTC/CNR, Italy

Dan Corkill	University of Massachusetts Amherst, USA
Daniel Moldt	University of Hamburg, Germany
Eric Matson	Purdue University, USA
Eric Platon	National Institute of Informatics, Japan
Eugénio Oliveira	Universidade do Porto, Portugal
Fuyuki Ishikawa	National Insitute of Informatics, Japan
Guido Boella	University of Turin, Italy
Harko Verhagen	Stockholm University, Sweden
Jaime Simão Sichman	University of São Paulo, Brazil
Javier Vázquez-Salceda	Universitat Politecnica de Catalunya, Spain
John-Jules Meyer	Utrecht University, The Netherlands
Juan Antonio R. Aguilar	IIIA-CSIC, Spain
Julian Padget	University of Bath, UK
Leon van der Torre	University of Luxembourg, Luxembourg
Liz Sonenberg	University of Melbourne, Australia
Luca Tummolini	ISTC/CNR, Italy
Marina de Vos	University of Bath, UK
Nicoletta Fornara	University of Lugano, Switzerland
Olivier Gutknecht	LPDL, France
Pablo Noriega	IIIA-CSIC, Spain
Pinar Yolum	Bogazici University, Turkey
Sascha Ossowski	URJC, Spain
Stephen Cranefield	University of Otago, New Zealand
Vincent Louis	Orange Labs, France
Virginia Dignum	University of Utrecht, The Netherlands
Viviane Torres Da Silva	Universidad Complutense de Madrid, Spain
Wamberto Vasconcelos	University of Aberdeen, UK

COIN@AAAI 2008

Alexander Artikis	NCSR Demokritos, Greece
Guido Boella	University of Turin, Italy
Olivier Boissier	ENS Mines Saint-Etienne, France
Rosaria Conte	CNR and University of Siena, Italy
Ulisses Cortes	UPC, Spain
Shaheen Fatima	University of Liverpool, UK
Mario Verdicchio	University of Bergamo, Italy
Nicoletta Fornara	University of Lugano, Switzerland
Scott Harmon	Kansas State University, USA
Henry Hexmoor	Southern Illinois University, USA
Koen Hindriks	Delft University of Technology, The Netherlands
Jomi Fred Hübner	ENS Mines Saint-Etienne, France
Christian Lemaître	Universidad Autónoma Metropolitana, Mexico
Victor Lesser	University of Massachusetts, USA
Pablo Noriega	IIIA-CSIC, Spain

James Odell	James Odell Associates, USA
Javier Vázquez-Salceda	Universitat Politecnica de Catalunya, Spain
Andrea Omicini	DEIS Universit di Bologna, Italy
Sascha Ossowski	University Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Juan Antonio R. Aguilar	IIIA, Spain
Paul Scerri	Robotics Institute-Carnegie Mellon University, USA
Jaime Simão Sichman	University of São Paulo, Brazil
Maarten Sierhuis	RIACS-NASA, USA
Catherine Tessier	ONERA-CERT, France
Walt Truszkowski	NASA, USA
Wamberto Vasconcelos	University of Aberdeen, UK

Additional Reviewers

Birna van Riemsdijk	Maite López
Francesco Viganò	Matteo Vasirani
Henrique Lopes Cardoso	Matthias Wester-Ebbinghaus
Huib Aldewereld	Pilar Dellunde
Luciano Coutinho	Ramon Hermoso Traba
Luis Paulo Reis	Serena Villata

Table of Contents

IV Design of Norms and Institutions

V Applications

Agreeing on Institutional Goals for Multi-agent Societies

Dorian Gaertner[1,2], Juan Antonio Rodríguez-Aguilar[2], and Francesca Toni[1]

[1] Dept. of Computing, Imperial College London, London SW7 2AZ, UK
{dg00,ft}@doc.ic.ac.uk
[2] IIIA-CSIC, Campus de la UAB
08193 Bellaterra, Spain
{dorian,jar}@iiia.csic.es

Abstract. We present an argumentation-based approach to the problem of finding a set of institutional goals for multi-agent systems. The behaviour of the autonomous agents we consider in this paper is goal-directed, driven by either individual or common goals. When multiple agents want to set up a collaboration framework (for themselves or others to use), they do so by forming an institution (or organisation). The goals of such institution must be agreed upon by the agents setting up the framework before it can be executed.

We propose to use argumentation, and in particular assumption-based argumentation, to facilitate the negotiation of institutional goals. We first describe a centralised approach and then provide the rationale for and detail our preliminary efforts at de-centralising the problem. We propose to employ the argumentation system CaSAPI as a tool to reason about the collaborative goals of the institution. Our approach mitigates concerns about performance bottlenecks and vulnerability of the system while providing, to some extent, privacy to the individual members of the institution.

1 Introduction

One of the concerns of multi-agent systems (MAS) research is how to achieve certain collective properties despite individual agents' varying behaviour. Thus, there is a wealth of approaches that consider how to get agents (with individual preferences and goals) to interact in such a way that their interactions lead to the desired global properties. Along these lines, economic-based approaches (e.g. coalition formation [25] and mechanism design [7]), cooperation-based approaches (e.g. teamwork [27]), and organisation-based approaches (e.g. organisations [8,13] and institutions [2]) provide MAS designers with techniques to enact MAS aimed at achieving their global goals. Notice that most approaches share the implicit assumption that there is some designer in charge of choosing the interaction mechanism that agents with individual goals use so that some global goals (or properties) are reached. Consider now that instead of a MAS designer, a group of agents gather together to decide by themselves the interaction

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 1–16, 2009.

rules of a MAS where they, or other agents, are to operate in. In fact, what we envisage is that a subset of all agents agrees upon the rules and the goals that constitute a regulatory environment (an institution or virtual organisation, see e.g. [23]). Once such an agreement is reached, these agents may or may not be part of the MAS they agreed upon while other agents that were not part of the subset of agents that agreed upon the rules of the society may join said society.

One approach to structuring the interaction between agents recently proposed is the notion of Electronic Institutions [2]. There, speech acts are considered as actions and special institutional agents control what can be uttered. The construction of such an institution and required institutional agents begins with the definition of the goals the institution is meant to achieve. In this paper, we focus on agent institutions defined as software environments composed of autonomous agents that interact according to predefined conventions on language and protocol, guaranteeing that certain norms of behaviour are enforced. An electronic institution is in a sense a natural extension of the social concept of institutions as regulatory systems which shape human interactions.

We will not describe how such an institution is designed or executed (amongst others, Esteva et al. [2] provide information on tools for such purposes and Castelfranchi investigates social power [5]), or how norms and normative positions are handled (see [14] for information on this subject). Instead, we will focus on an earlier stage of the development of such institutions, namely on the question of how multiple agents can join efforts and agree on institutional goals. Such agents would still have individual goals, in addition to their common, institutional goals[1] for a particular collaboration. It is further worth noting that we begin with the supposition that the agents *want* to form an institution. Prior to entering into the argumentation process that we describe in this paper, they will need to explore whether or not they want to collaborate and form an institution at all. We assume that they have answered that question affirmatively.

We will describe two ways of constructing the set of common, institutional goals, both employing the CaSAPI tool [15] for assumption-based argumentation [4]. Firstly, a centralised approach is presented that combines the different goals of all agents and all their individual knowledge bases in the best possible way. Secondly, we detail an approach where each agent expresses its preference or rejection of a goal. A mechanism is then presented where participating agents use arguments based on their individual knowledge to defend their position.

This paper is structured as follows: Section 2 describes assumption-based argumentation as well as the CaSAPI tool and Section 3 introduces an example scenario. We then present the centralised approach in Section 4, show how CaSAPI can realise this approach (in the context of the scenario), discuss the issue of control and describe the disadvantages of the centralised approach before detailing preliminary work on a distributed argumentation mechanism to find common goals for the institution in Section 5. Finally, we look at related work and conclude.

[1] We will assume that the desired collaboration will take place in an institution and hence we will equate collaboration goals and institutional goals in what follows.

2 Assumption-Based Argumentation

This section provides the basic background on assumption-based argumentation (ABA) and the CaSAPI tool, see [4,10,11,12,15,16] for details.

An ABA framework is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where

- $(\mathcal{L}, \mathcal{R})$ is a *deductive system*, consisting of a language $\mathcal{L}$ and a set $\mathcal{R}$ of inference rules,
- $\mathcal{A} \subseteq \mathcal{L}$, referred to as the set of *assumptions*,
- $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{x}$ is referred to as the *contrary* of x.

Intuitively, inference rules may be domain-specific or domain-independent [4]. They may correspond to causal information, to inference rules and axioms in a chosen logic-based language [4] or to laws and regulations [24]. Assumptions are sentences that can be questioned and disputed (as opposed to axioms that are beyond dispute), for example uncertain or unsupported beliefs or decisions. The contrary of an assumption, in general, stands for a reason why the assumption may be undermined (and thus may need to be be dropped).

We will assume that the inference rules in $\mathcal{R}$ have the syntax $l_0 \leftarrow l_1, \ldots l_n$ (for $n \geq 0$) where $l_i \in \mathcal{L}$. We will represent the rule $l \leftarrow$ simply as l. As in [10,12,15,16,11], we will restrict attention to *flat* ABA frameworks, such that if $l \in \mathcal{A}$, then there exists no inference rule of the form $l \leftarrow l_1, \ldots, l_n \in \mathcal{R}$, for any $n \geq 0$. These frameworks are still quite general and admit many interesting instances [4]. Furthermore, we will adopt a generalisation of ABA frameworks, first given in [15], whereby assumptions allow *multiple contraries*, i.e.

- $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\wp(\mathcal{L})$.

As discussed in [11], multiple contraries are a useful generalisation to ease representation of ABA frameworks, but they do not really extend their expressive power.

Given an ABA framework, an *argument* in favour of a sentence $x \in \mathcal{L}$ supported by a set of assumptions X, denoted $X \vdash x$, is a (backward) deduction from x to X, obtained by applying backwards the rules in $\mathcal{R}$.

Example 1. Let us consider the following ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where:

$\mathcal{L} = \{p, a, \neg a, b, \neg b\}$,
$\mathcal{R} = \{p \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a\}$,
$\mathcal{A} = \{a, b\}$ and
$\overline{a} = \{\neg a\}$, $\overline{b} = \{\neg b\}$.

Then, an argument in favour of p supported by $\{a\}$ may be obtained by applying $p \leftarrow a$ backwards (the argument is $\{a\} \vdash p$).

In order to determine whether a conclusion (set of sentences) should be drawn, a set of assumptions needs to be identified providing an "acceptable" support for the conclusion. Various notions of "acceptable" support can be formalised,

using a notion of "attack" amongst sets of assumptions whereby X_1 *attacks* X_2 iff there is an argument in favour of some $y \in \overline{x}$ supported by (a subset of) X_1 where x is in X_2 (for example, given $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ above, $\{b\} \vdash \neg a$ attacks $\{a\} \vdash p$). Then, a set of assumptions is deemed

- *admissible*, iff it does not attack itself and it counter-attacks every set of assumptions attacking it;
- *complete*, iff it is admissible and it contains all assumptions it can defend, by counter-attacking all attacks against them;
- *grounded*, iff it is minimally (w.r.t. set inclusion) complete;
- *ideal*, iff it is admissible and contained in all maximally (w.r.t. set inclusion) admissible sets.

All these notions are possible formalisations of the notion of "acceptable" support for a conclusion. The first is a *credulous* notions, possibly sanctioning several alternative sets as "acceptable" supports, the latter two are *sceptical* notions, always sanctioning one single set as "acceptable" support. Different computational mechanisms can be defined to match these notions of "acceptable" support for given claims.

The CaSAPI system that we propose to use in this paper (CaSAPI version 2 [15]) allows to compute the computational mechanisms of GB-dispute derivations for computing grounded supports [12], AB-dispute derivations for computing admissible supports [10,12] and IB-dispute derivations for computing sceptical supports [11,12]. In the case of example 1, there is an AB-dispute derivation for the claim p, computing the admissible support $\{a\}$. However, GB- and IB-dispute derivations fail to find grounded and ideal supports (respectively) for p, since indeed p cannot be sceptically supported (as $\{a\}$ and $\{b\}$ are "equally good" alternative sets of assumptions). If $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ in Example 1 is modified so that both c and $\neg c$ are added to $\mathcal{L}$ and the last inference rule in $\mathcal{R}$ is replaced by the two rules:

$$\neg b \leftarrow c;\ \neg c$$

with c an additional assumption and $\overline{c} = \{\neg c\}$, then there exist AB-, GB- and IB-dispute derivations for the claim p, all computing the support $\{a\}$ (which is now admissible, grounded and ideal).

Figure 1 illustrates how the rules, assumptions and contraries of the simple ABA framework modifying Example 1, as given earlier, are entered into CaSAPI using a graphical user interface. The user can enter a claim to be proved (p in what follows), select the type of dispute derivation it requires and control various other features of the computation such as the amount and format of the system's output. Once the input is entered, CaSAPI can be `Run` to determine whether of not the claim admits an "acceptable" support, according to the chosen type of dispute derivation. We will use CaSAPI with GB-dispute derivations only, as this semantics best fits the needs of our application (cf. Section 4 for details).

Note that, compared to Dung's conventional abstract argumentation [9], ABA addresses three problems: (i) how to find arguments, (ii) how to determine attacks and (iii) how to exploit the fact that different arguments may share premises.

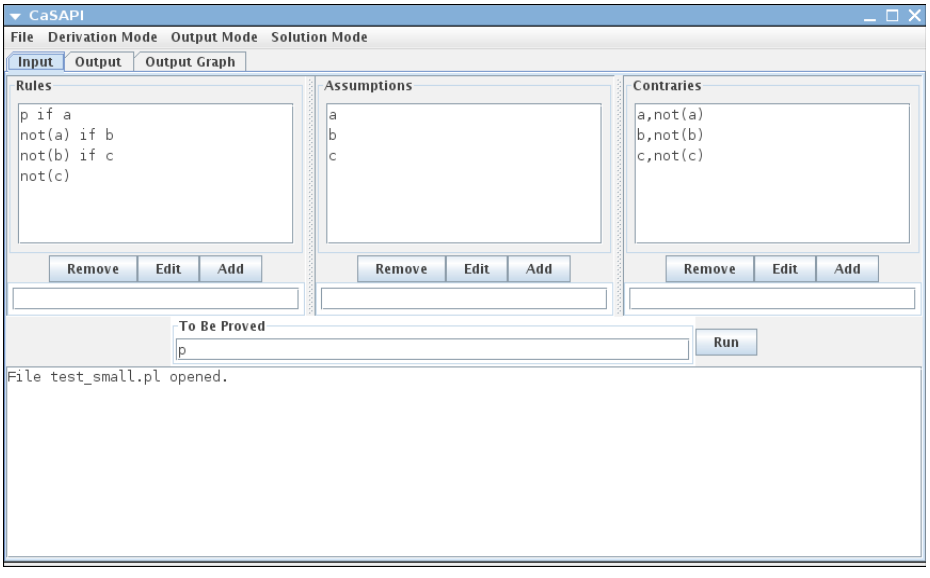


Fig. 1. Screenshot of the input process to CaSAPI

These problems are ignored by abstract argumentation, that sees arguments and attacks as "black-boxes". Instead, in ABA arguments are defined as *backward deductions* (using sets of *rules* in an underlying logic) supported by sets of *assumptions*, and the notion of attack amongst arguments is reduced to that of *contrary* of assumptions. Moreover, (iii) is addressed by all forms of dispute-derivations (AB-, GB- and IB-) by employing various forms of "filtering steps" (for details, see [10,12]).

ABA can be seen as an instance of abstract argumentation, but is general-purpose nonetheless (in that, e.g., it can be instantiated in many ways to get many different frameworks for non-monotonic reasoning [4]). The relationship between ABA and abstract argumentation is detailed in [12] and exploited by versions 3 and 4 of CaSAPI [16,11]: we ignore these versions here because they only support AB-dispute derivations and we chose the GB-dispute derivations for our application (cf. Section 4 for details).

3 Scenario

Reaching agreements in a society of self-interested agents is a problem related to the issue of cooperation. The capabilities of negotiation and argumentation are paramount to the ability of agents to reach agreements but as far as we know, neither negotiation-based nor argumentation-based approaches have been explored for the design of institutions. The automated design of (market) institutions has relied so far on mechanism design, namely on the design of protocols for governing multi-agent interactions that have certain desirable properties. However,

the use of mechanism design for some (non-financial) institutions where there is a need for either justifying or changing negotiation stances is not sufficient.

Nobel laureate Ronald Coase in 1937 noted that in order to "make things", a collaboration is required and setting up such collaboration is costly. He used this insight to justify the existence of big companies which otherwise would be replaced by individuals operating in free and unregulated markets. In his recent book [28], Don Tapscott describes how in an increasingly connected world, the cost of collaborating is evaporating and the raison d'être of huge corporations ceases to exist. He provides the following example[2] to illustrate his claim:

> Take the Chinese motorcycle industry, which has tripled its output to 15m bikes per year over the past decade. There aren't really any Chinese equivalents of the big Japanese and American firms - Honda or Harley. Instead, there are hundreds of small firms [...]. Their representatives meet in tea-houses, or collaborate online, each sharing knowledge, and contributing the parts or services they do best. The companies that assemble the finished products don't hire the other companies; assembling the finished product is just another service. A "self-organised system of design and production" has emerged [...].

One can easily envisage that such ad-hoc collaborations will have a number of generic goals such as good communication between collaborators, sustainability of the collaboration (or the goal of achieving the objectives by a certain deadline), quality leadership and a high degree of connectedness and unity between the different participants. On a less generic dimension, business metrics can serve as goals for the institution, too. For example, profit increases, reduction in risk and rise of business value of the collaboration as a whole and/or of its participants can be considered collaboration goals. Finally, domain (i.e. collaboration) specific goals need to be considered such as the production of a certain amount of goods, the maintenance of a certain level of employee satisfaction, a certain throughput, a certain penalty on delivery delays and the return on capital employed.

Not all of these goals will be explicit goals of any given institution and the collaborators can decide which ones they value most and should strive to embed in their operational processes. If all collaborators have the same goals and they are all achievable (i.e. the set of goals is consistent and acceptable) then this set of goals will be the basis for the construction of the institution. However, in most cases, individual collaborators have conflicting ideas of the goals of the collaboration and as these goals must be shared by all collaborators, a mechanism is needed that reaches an agreement. We present a solution to this problem based on ABA.

In order to demonstrate our approach in this paper, we use a scenario of three agents named Adrian, Betty and Carles, working in the Chinese motorcycle industry and intending to institutionalise a collaboration for themselves to operate. They share a common language $\mathcal{L}$ in order to avoid ontological

[2] The example is quoted from a review of the book which appeared in the Guardian newspaper on September 5th, 2007.

misunderstandings[3]. These three agents have some shared knowledge between them (stored in the shared knowledge base SKB) and some individual knowledge (stored in IKB_A, IKB_B and IKB_C, respectively). Each agent has some goals stored in its personal goal base (in this example, $GB_A = \{g_1, g_2, g_3\}$, $GB_B = \{g_2, g_4, g_5\}$ and $GB_C = \{g_1, g_6\}$) that it would like to see as the goals of the collaboration, and an individual vision (represented as rules in the corresponding IKB) of how these goals can be achieved (i.e. what is required for them to be reached). Each one wants its own goals to become common goals of the institution.

g_1: *to produce 100 motorbikes this week*
g_2: *to always clear the assembly line at the end of each day*
g_3: *to produce 100 sidecars this week*
g_4: *to improve/foster relations between the three collaborators*
g_5: *to make the institution sustainable/repeatable*
g_6: *to make Carles as the leader of this collaboration*

Whether or not a goal is achievable depends on a number of facts to hold true and a number of assumptions to be "assumable". The individual visions of how to achieve (relevant) goals are given in Table 1. Here the IKBs are represented as sets of rules of ABA frameworks. To ease understanding, the rules for each agent are partitioned into rules for achieving goals and rules for propositions and beliefs needed to support goals. The CaSAPI tool that we propose to employ treats all rules in the same way. Furthermore, it is a coincidence that each agent i has goal rules for exactly the goals in its corresponding GB_i as in general agents may hold information about how to achieve goals they do not hold themselves.

Table 1. Example scenario as ABA framework

IKB_A	IKB_B	IKB_C
$g_1 \leftarrow a_1, x_1$ $g_2 \leftarrow p_1, x_2$ $g_3 \leftarrow p_2, a_2, x_3$ $g_3 \leftarrow a_3, x_4$	$g_2 \leftarrow q_1, x_5$ $g_4 \leftarrow b_1, q_2, x_6$ $g_5 \leftarrow q_3, b_2, x_7$	$g_1 \leftarrow r_1, c_1, x_8$ $g_6 \leftarrow r_2, x_9$
p_1 $p_2 \leftarrow a_1$	$q_1 \leftarrow b_3$ q_2 q_3 $\neg b_2$	r_3 $\neg a_3 \leftarrow r_3$ $\neg b_1 \leftarrow r_4$ $r_4 \leftarrow r_5, c_2$ $r_5 \leftarrow c_3$

For example, goal g_1 can be achieved in two ways — it can be achieved assuming *sufficiently many spare parts are in stock* (assumption a_1) or it can be achieved if *a reliable third part producer exists* (proposition r_1) and it can be assumed that *outsourcing is acceptable to all collaborators* (assumption c_1). Another example is that goal g_4 can be achieved when b_1 can be assumed (*everyone*

[3] In the future, ontology mapping methods (see e.g. [21]) can be used to align different languages.

is happy) and proposition q_2 holds (*the collaboration is profitable as a whole*). Finally[4], goal g_5 is achieved if *all collaborators are trustworthy* (proposition q_3) and assuming there is *at least constant demand for motorcycles* (assumption b_2).

Betty's individual knowledge base IKB_B would therefore contain the rule $g_4 \leftarrow b_1, q_2$ as well as two other goal rules. In what follows, we treat all p_i, q_i and r_i as propositions (i.e. elements of a propositional language $\mathcal{L}$ that are not assumptions) and all a_i, b_i and c_i as assumptions for Adrian, Betty and Carles, respectively. Assumptions can represent beliefs one cannot prove or disprove or actions to be performed by an agent. The choice is left to the designer of the agents. In addition to these a_i, b_i and c_i's, there are so-called applicability assumptions that we denote with x_i which are attached to each goal rule. Hence a rule $g_6 \leftarrow r_2, x_9$ can be read as "goal g_6 (about *Carles as leader*) can (usually) be achieved if proposition r_2 (*Carles is the most senior agent*) holds" and this rule is applicable if x_9 is the case. An agent can dispute the applicability of this rule (and thereby argue that g_6 can not be achieved in this way) by showing the contrary of x_9 (e.g. that seniority is irrelevant when determining the leader of a collaboration).

In addition to the individual knowledge bases, there is a shared knowledge base SKB containing the fact that *not sufficiently many spare parts are in stock* ($\neg a_1$) and the rule $r_1 \leftarrow r_3$, stating that *a reliable third party producer exists* if *MotoTaiwanInc has been reliable in the past.*

When defining an ABA framework one must also specify the contraries of all assumptions. For simplicity, we take the notion of contrary to be logical negation. Therefore, we have $\overline{x} = \neg x$ for any assumption x. We can now define an ABA framework for Betty where the inference rules are $\mathcal{R}_B = IKB_B \cup SKB$, the set of assumptions is $\mathcal{A}_B = \{b_1, b_2, b_3, x_5, x_6, x_7\}$ and the contrary of each assumption is its logical negation. The language $\mathcal{L}_B$ is made up of all literals that feature in the rules in IKB_B.

Using this ABA framework, Betty on her own can see that goal g_2 is achievable, if b_3 can be assumed (since b_3 *'the deal with the external distributor is fair'* allows to deduce q_1 *'a working distribution channel exists'* and that proposition is needed for goal g_2). Furthermore, she can achieve g_4 if b_1 can be assumed. Since $\neg b_2$ is a fact that Betty knows about, she can already see on her own that g_5 is not an acceptable goal for the collaboration and will thus never put it forward. We can similarly construct ABAs $\langle \mathcal{L}_A, \mathcal{R}_A, \mathcal{A}_A, \overline{\ } \rangle$ and $\langle \mathcal{L}_C, \mathcal{R}_C, \mathcal{A}_C, \overline{\ } \rangle$ for Adrian and Carles, respectively. Note, for each $i \in \{A, B, C\}$, $\mathcal{L}_i \subseteq \mathcal{L}$.

We now look at two approaches to determine the set of common goals between the three agents that can then be used to construct an institution from them.

4 Centralised Approach

Any agent-based institution requires a set of institutional goals which are used to create the structure and elements of the institution. When human designers

[4] We refrain from detailing how to achieve the remaining goals and leave it to the reader's imagination to fill the other elements of $\mathcal{L}$ (e.g. p_2 and a_3) with meaning.

dictate these goals, the institution can be constructed from them. However, when several autonomous agents come together to form an institution, they must agree on a set of institutional goals that are acceptable to all of them. Note that within ABA a goal is "acceptable" with respect to a given semantics if there exists an acceptable set of assumptions according to that semantics supporting the goal.

Considering the semantics described in Section 2, we propose to use the sceptical grounded semantics for our scenario. As argued in [12], some domains such as legal reasoning (where a guilty verdict must be obtained via sceptical reasoning) and multi-agent systems (where decisions must be made unanimously) require sceptical semantics. In this paper we propose that agents argue about goals for their future collaboration and such goals must be agreed upon scepically — the set of acceptable goals must be unique. As justification consider the case where two goals depend upon two conflicting assumptions. Credulous semantics would allow to find support for either goal individually while sceptical reasoning excludes both goals[5]. Using the former, the acceptance of goals would hence be dependent on the order in which they are considered — which is a complication we will leave for future work.

Assume n agents want to collaborate. An intuitive approach to find a set of institutional goals consists of combining all the individual knowledge bases IKB_i of the n agents with the shared knowledge base SKB and reason with this combined knowledge. We would have the following assumption-based argumentation framework:

$\mathcal{R} = SKB' = SKB \cup \bigcup_{k=0}^{n} IKB_k$

$\mathcal{A} = \bigcup_{k=0}^{n} \mathcal{A}_k$ where $\mathcal{A}_k$ are the assumptions of agent k including its applicability assumptions.

The contrary of an element x of $\mathcal{A}$ can be constructed by using the fact that the individual sets of assumptions are disjoint, as follows: find the agent i that has x as an assumption and use the contrary function of agent i. If the requirement that two sets $\mathcal{A}_k$ are pair-wise disjoint is dropped, the combined contrary function $\overline{}'$ is constructed as follows:

$\overline{x}' = \{y \mid y = \overline{x}^i \text{ and } i \text{ is an agent } \}.$

This combined contrary function returns sets of elements. For example, if Betty and Carles have the assumption α that the sky is blue and Betty thinks that $\overline{\alpha} = \{sky\_is\_grey\}$ while Carles thinks that $\overline{\alpha} = \{sky\_is\_red\}$, then the combined system would return as a set of contraries for the assumption α the set $\{sky\_is\_grey, sky\_is\_red\}$. Showing that any one of these contraries holds, is sufficient to disprove α. Whether or not the sets of assumptions are disjoint depends on the kind of institution required.

Having constructed such an ABA framework, we can now run the argumentation system CaSAPI and query it about one goal at a time. Those goals that

[5] CaSAPI also supports the *ideal* semantics, which is less sceptical than the grounded one. It also has a unique extension but the set of goals it accepts is a superset of those accepted using grounded semantics. In future work, we will evaluate the differences.

are acceptable according to the chosen semantics (see Section 2) are selected as institutional goals.

In the example scenario with six individual goals, only goals g_1 and g_2 will become institutional goals. Adrian's rule $g_1 \leftarrow a_1$ is not helpful, since $\neg a_1$ is part of the shared knowledge base, but Carles has a way of showing g_1 provided c_1 can be assumed. CaSAPI attempts (and fails) to use the rule $g_1 \leftarrow a_1$ before backtracking and succeeding by using Carles' way of showing g_1. Goal g_2 is acceptable to Betty (as discussed in Section 3.1) and since nobody can attack her argument (that from assumption b_3 the goal is possible), it becomes an institutional goal. Note, however, that Betty's goal g_4, which was acceptable to her before, given that b_1 can be assumed, is not an institutional goal. The reason is that Carles has a way of showing $\neg b_1$ (*not everyone is happy*) provided he can assume c_2 and c_3. Since between the three agents nobody can attack either c_2 or c_3, Carles' attack succeeds and Betty's goal g_4 is not acceptable.

4.1 Control Issues

Rather than using CaSAPI to query one potential goal at a time, we can also use a meta-interpreter which will attempt to show that all goals are acceptable at once. If this fails, the meta-interpreter will remove one goal at a time from the set of all goals (and possibly backtrack) in order to find the biggest acceptable subset of goals. If a reason is put forward to use a different semantics (e.g. admissibility), then the meta-interpreter will need to make use of additional machinery to control the order of querying. Finally, agents could express a preference of the goals they would like to see adopted as goals of the collaboration. Again, the meta-interpreter will have to handle these. We leave the construction of such a meta-interpreter for future work.

4.2 Disadvantages

Applying the CaSAPI tool in a centralised manner, while being a computationally straight-forward approach, comes with several disadvantages. Combining the individual knowledge bases can lead to performance issues, since all the computational burden needs to be carried by one central entity which computes the optimal set of institutional goals. This agent (and the goal finding process) will quickly become the bottleneck of the system. It also increases the system's vulnerability to attacks, since without this entity's capability to execute the centralised algorithm, the agents cannot continue their efforts to form a society. Furthermore, this central agent needs to be trusted, which brings with it even more challenges.

However, the biggest disadvantage of the centralised approach concerns *privacy*. The individual agents will have to share their individual rules, knowledge and other internal details that they may want to keep secret from each other. For example, if Adrian, Betty and Carles want to form a market-place institution then while they need some common goals to make their venture happen, they would be forgiven for being hesitant to share all their business knowledge (such as detailed business rules) with one another.

5 Distributed Approach

In order to allow the participating agents some privacy and to keep their individual rules private, we propose a distributed approach which does away with a central entity which amalgamates information. Instead, each of the agents needs to be equipped with the CaSAPI engine in order to compute arguments and their supporting assumptions. Communication between the agents will be used to distribute arguments and attacks against these arguments are again computed locally. The only prerequisite we insist on is a shared understanding of the notion of contrary.

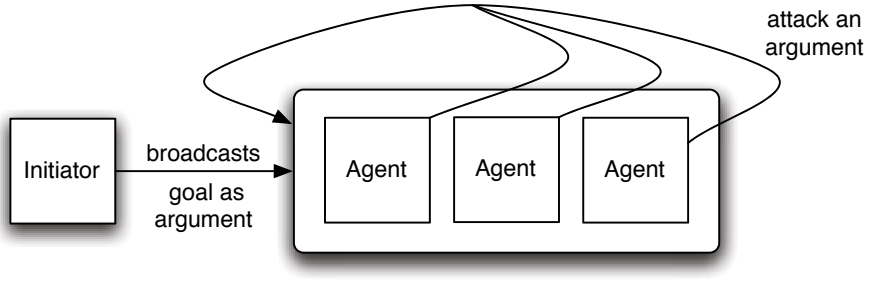


Fig. 2. Schematic description of interaction protocol

Each single goal will be treated separately, in a new conversation. Each conversation starts with one agent, the initiator whose individual goal is concerned, broadcasting a message with the goal in question and the supporting assumptions needed to defend this goal.

Every agent receiving this message then attempts to find an attack by looking for some support for the contrary of one of the assumptions in the initial message. This includes disputing one of the rules used to build the argument by showing the contrary of the applicability assumption. An agent who finds such an attack, broadcasts it (together with the assumptions needed to defend it). Everyone is then trying to counter-attack this attack (again by searching for supporting assumptions for an argument in support of a contrary of an assumption of the attack). This collaborative process implicitly constructs a tree of arguments and continues until no more attacks can be found and the initial argument either prevails or is defeated. If it prevails, it becomes an institutional goal.

This process can be clarified with an example from the running scenario. Imagine Betty initiates a new conversation by broadcasting her goal g_4 with the supporting set of assumptions $\{b_1\}$. Now Adrian and Carles both attempt to find an argument in favour of $\neg b_1$ since this is the only possibility to attack Betty's argument. Adrian is unsuccessful, but Carles finds an attack, namely an argument in favour of $\neg b_1$ supported by $\{c_2, c_3\}$. Since neither Adrian nor Betty can find arguments for either of $\neg c_2$ or $\neg c_3$, Carles' attack prevails and Betty's initial argument is defeated. Note that Adrian may withhold an attack consciously in order to help Carles. We leave the issue of collusion for future work.

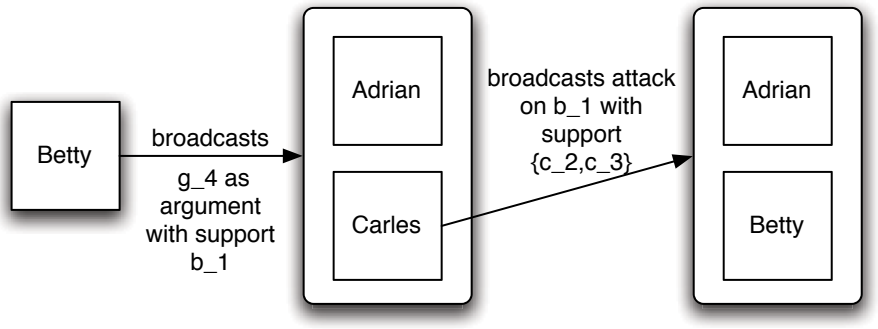


Fig. 3. An instance of the interaction protocol from the scenario

Each individual agent only initiates conversations about goals that it considers possible. Therefore, Betty will not propose goal g_5 to the other agents, as she herself is able to show its impossibility. Hence agents will never attack their own proposals. Further implementation issues are discussed in Section 5.2 of this paper but first we summarise the advantages of the distributed approach.

5.1 Advantages

The advantages of this distributed approach are three-fold:

Less vulnerability of the system as a whole, since even if an agent fails to perform (e.g. is shut down), the other agents can still continue to look for an agreement.

Privacy is maintained to the extent that rules in the individual knowledge bases are not shared between agents. In the example above where Carles successfully attacked Betty's argument, he did not have to reveal his rule $r_4 \leftarrow r_5, c_2$, for example. This privacy is useful but requires that the agents are honest. If this is not the case, an agent could counter-attack any attack on his proposed goals with a fictional support set.

Efficiency is gained in two ways. ABA provides computational savings via several filtering mechanisms (all of which employed in CaSAPI). Additionally, each agent can locally store the successful and unsuccessful arguments from the dialectical structure that is computed each time the argumentation system is run. These stored arguments can then be re-used in future conversations to save recomputing their support sets. Some of these savings will however be offset by increased communication cost. A detailed investigation into the comparative costs of the two approaches is part of our plan for future work.

5.2 Implementation Issues

We want to briefly touch upon two issues that require further discussion. The first one is the *order of goal proposals*. If a no meta-interpreter (cf. Section 4.1) is

used, then in the simplest case, a token-based approach can be employed where the n agents that want to reach an agreement form a circle and only the agent in possession of the token can initiate a new conversation. After the initiation it passes the token on to the next agent in clock-wise direction. An agent can also pass the token on without initiating a new conversation, if all his goals have been discussed or are in discussion. If the token moves n times without a new conversation being started, then the process finishes.

This simple approach can be optimised in various ways that we do not want to go into too deeply here. Suffice to say the order in which the goals are considered, while not changing the final result[6], has an impact on efficiency, since due to the storing of (sub-)arguments described in Section 5.1 conversations can be shortened significantly.

A second issue concerns the termination of a conversation. Above we said that the arguing stops when no more attacks can be found. We consider two ways in which this is implemented. If all agents operate on the same (or sufficiently similar) clock, a time-out mechanism can be used. If no attack has been broadcast within a specified number of seconds of the initiation of the conversation (or of the broadcasting of the most recent attack), then the argument (or the attack) prevails. A more elaborate approach has the agents explicitly stating that they cannot find an attack on a given set of supporting arguments. It involves a conversation protocol including messages to that effect. Further work is needed to formalise these protocols.

6 Related Work and Conclusions

In this paper we present original but preliminary work on the problem of finding a set of institutional goals for multi-agent systems from which institutions can be constructed. Research on agent organisations and institutions has mainly focused on specification languages (e.g. [2]), architectures and software tools and frameworks (e.g.[2,18,19]), agent reasoning (e.g. [22]), and understanding the evolution of norms [3]. Somewhat related approaches are found in [1] and [26], but to the best of our knowledge, no efforts on automating institutional design, from the conception of goals to the enactment of the rules, have been carried out. In this paper we take the first steps along this direction.

We are proposing to use assumption-based argumentation [4,9,15] and have lined-out two approaches to the problem of determining a set of institutional goals. One may argue that institutional goals should be more general and abstract than the goals of individual agents. For example, from the personal goal *"I want to finish negotiating by 8pm"*, one can deduce the institutional goal *"We should all finish by 8pm"*. We plan to investigate this in the future. For this paper we assume the individual goals are sufficiently general (i.e. as the institutional goal above).

[6] When a credulous semantics is used, such as admissibility, correctness does become an issue. For the GB-dispute derivations we use in this paper, the order in which the goals are considered has no impact on correctness.

A second line of investigation involves introducing trust and reputation into the argumentation and interaction mechanism. One could place more importance on the arguments of certain agents and then use a preference-based approach to resolve conflicts between goals as well as arguments (we took a related approach for solving conflicts between norms in [17]). Another interesting notion is favouritism (i.e. not attacking an argument even though one could).

Finally, one can extend the same process to reasoning about joint norms. Some norms may be derived from institutional goals, others could be agreed upon using a similar approach to the one sketched in this paper.

A somewhat related field is that of team formation. An agent team consists of a number of cooperative agents which have agreed to work together toward a common goal [20]. The challenges associated with team formation involve determining how agents will be allocated to address the high-level problem, maintaining consistency among those agents during execution, and revising the team as the environment or agent population changes. In our case we focus on the negotiation that occurs prior to the formation process, namely on the agreement of high-level goals.

The work on joint intentions [6] can also be seen as relevant although it has primarily focused on understanding the motivations for a team of agents to jointly pursue/drop goals. Thus, the main focus has been on understanding cooperation as a collective (intentional) decision-making process that makes agents adopt joint actions. The working assumption is that collective intentional behaviour cannot be analysed in terms of individual intentions. In our case, we are not concerned on agents' collective mental state or decision-making, but on the argumentation machinery employed to eventually reach a collective agreement.

References

1. Amgoud, L., Kaci, S.: An argumentation framework for merging conflicting knowledge bases. International Journal of Approximate Reasoning 45, 321–340 (2007)
2. Arcos, J.L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Engineering open environments with electronic institutions. Engineering Applications of Artificial Intelligence 18(2), 191–204 (2005)
3. Axelrod, R.: An evolutionary approach to norms. The American Political Science Review 80(4) (1986)
4. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic framework for default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)
5. Castelfranchi, C.: Social power: A point missed in multi-agent dai and hci. Decentralized A.I., 49—62 (1990)
6. Cohen, P.R., Levesque, H.J., Smith, I.: On team formation. In: Hintikka, J., Tuomela, R. (eds.) Contemporary Action Theory. Synthese (1997)
7. Dash, R.K., Jennings, N.R., Parkes, D.C.: Computational-mechanism design: A call to arms. IEEE Intelligent Systems 18(6), 40–47 (2003)
8. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. PhD thesis, Utrecht University (2003)

9. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77 (1995)
10. Dung, P.M., Kowalski, R.A., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence 170, 114–159 (2006)
11. Dung, P.M., Mancarella, P., Toni, F.: A dialectic procedure for sceptical, assumption-based argumentation. In: Proceedings of the First International Conference on Computational Models of Argument, Liverpool (2006)
12. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence - Special Issue on Argumentation in Artificial Inelligence 171(10-15), 642–674 (2007)
13. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
14. Gaertner, D., Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J.A., Vasconcelos, W.: Distributed norm management in regulated multi-agent systems. In: Proceedings of the Sixth International Conference on Autonomous Agents and Multi-Agent Systems (2007)
15. Gaertner, D., Toni, F.: CaSAPI: a system for credulous and sceptical argumentation. In: Proceedings of the International Workshop on Argumentation and Non-Monotonic Reasoning (ArgNMR) (2007)
16. Gaertner, D., Toni, F.: Computing arguments and attacks in assumption-based argumentation. IEEE Intelligent Systems 22(6), 24–33 (2007)
17. Gaertner, D., Toni, F.: Preferences and assumption-based argumentation for conflict-free normative agents. In: Rahwan, I., Parsons, S., Reed, C. (eds.) Argumentation in Multi-Agent Systems. LNCS, vol. 4946, pp. 94–113. Springer, Heidelberg (2008)
18. Gâteau, B., Boissier, O., Khadraoui, D., Dubois, E.: MoiseInst: an organizational model for specifying rights and duties of autonomous agents. In: Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS), pp. 484–485 (2005)
19. Gutknecht, O., Ferber, J., Michel, F.: Integrating tools and infrastructures for generic multi-agent systems. In: Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS 2001), pp. 441–448 (2001)
20. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. Knowledge Engineering Review 19(4), 281–316 (2005)
21. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: The state of the art. Knowledge Engineering Review (2003)
22. López y López, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, pp. 674–681 (2002)
23. Patel, J., Luke Teacy, W.T., Jennings, N.R., Luck, M., Chalmers, S., Oren, N., Norman, T.J., Preece, A.D., Gray, P.M.D., Shercliff, G., Stockreisser, P.J., Shao, J., Alex Gray, W., Fiddian, N.J., Thompson, S.G.: CONOISE-G: agent-based virtual organisations. In: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, pp. 1459–1460 (2006)
24. Prakken, H., Sartor, G.: On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. In: Proceedings of the Fifth International Conference on Artificial Intelligence and Law, pp. 1–10. ACM Press, New York (1995)

25. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101(1-2), 165–200 (1998)
26. Sycara, K.: Argumentation: Planning other agents' plans. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (1989)
27. Tambe, M., Zhang, W.: Towards flexible teamwork in persistent teams: Extended report. Autonomous Agents and Multi-Agent Systems 3(2), 159–183 (2000)
28. Tapscott, D., Williams, A.D.: Wikinomics: How Mass Collaboration Changes Everything. Portfolio Hardcover (2006)

Organizations and Autonomous Agents: Bottom-Up Dynamics of Coordination Mechanisms

Bob van der Vecht[1,2], Frank Dignum[2], John-Jules Ch. Meyer[2], and Virginia Dignum[2]

[1] TNO Defence, Safety and Security
Oude Waalsdorperweg 63, 2597 AK, Den Haag, The Netherlands
bob.vandervecht@tno.nl
[2] Department of Information and Computing Sciences, Universiteit Utrecht
Padualaan 14, 3584 CH, Utrecht, The Netherlands
{dignum,jj,virginia}@cs.uu.nl

Abstract. Agents in an organization need to coordinate their actions in order to reach the organizational goals. Organizational models specify the desired behaviour in terms of roles, relations, norms and interactions. In this research, we show how organizational rules can be adopted by autonomous agents. We have developed a method to translate norms into reasoning rules for agents. However, since the agents are autonomous they will have their own reasoning rules next to the organizational rules. We propose a modular reasoning model to make organizational rules explicit. This allows for meta-reasoning about these rules. We show that this stimulates bottom-up dynamics in the organization.

1 Introduction

Organizations benefit from autonomous decisions by their participants. This is visible in human organizations. Not only the formal organizational structure but also the informal circuit of communication and interaction determines the success of an organization. In human organizations a participant's contribution is evaluated based on the organizational requirements as well as the extra achievements. Someone who takes initiative and builds up a personal network is often more appreciated than someone who sticks to the official rules and does not do anything extra. The capability to act and make decisions in unexpected situations, is usually perceived as a positive characteristic of human actors.

How does the observation that organizations benefit from participants' initiatives translate to multi-agent coordination that are based on organizational theory? Every organization is created for a specific objective. The organizational model describes the desired global behaviour using abstract concepts such as roles, relations and norms. Its specification is meant to guarantee certain requirements, for example, about the information flow. However, since the agent is assumed to be an autonomous entity, decision making is a local process of the agent. Therefore, it is important to maintain agent autonomy within the multi-agent coordination model. The organizational rules should guide the choices of the agent, but the organization cannot control the agent's decision-making process.

In this research, we investigate how to make agents aware of organizational rules. At the same time we allow them to take initiatives besides the formal structure. We

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 17–32, 2009.

propose a modular approach with which agents can adopt organizational rules into their reasoning model. The reasoning model separates the organizational rules from the actual decision-making process. This way, the agent's decision-making process can be defined separately from the coordination mechanism.

At the same time, the modular approach allows for meta-reasoning about different behavioural rules, which makes the agent independent from the organizational structure. The agent is not limited in its decision-making. It knows how to follow the organizational norms and it is able to take other initiatives. Therefore, the model guarantees agent autonomy.

The paper is structured as follows. In Sect. 2 we discuss related work on agent organizations. We motivate our choice to use the OperA model to describe organizations and in Sect. 3 we give an example of the use of OperA. Section 4 describes a reasoning model with which an agent can adopt organizational constraints to its decision making. In Sect. 5 we show how the organizational rules are adopted by the agent. In Sect. 7, we investigate bottom-up dynamics in organizations using the autonomy of agents and we give examples.

2 Agent Organizations: Related Work

Researchers in multi-agent systems have introduced the organizational metaphor to achieve coordination between autonomous agents. Organizational models specify coordination mechanisms between agents in abstract concepts, such as roles, relations and norms. In this section we discuss related work on organizational models. As agents are expected to be autonomous entities it is not straightforward to operationalize organizations. We describe how different approaches allow agents to take up organizational tasks. We motivate our choice to use OperA [1] for organizational description and we briefly explain the model.

2.1 Models for Agent Organizations

Several organizational descriptions have been proposed. One of the first was the Agent Group Role (AGR) model [2], that introduced the concepts of roles, groups and interaction between roles. The model focuses on defining the structural aspects of the organization. The interaction between agents is defined within the role description. The AGR model does not consider abstract behaviour rules, such as norms.

The OperA model [1] proposes a more expressive way for defining organizations by introducing an organizational model, a social model and an interaction model. This approach explicitly distinguishes between the organizational model and the agents that act in it. Agents become aware of the organizational rules via contracts that specify these rules. The agents are still fully autonomous in making decisions.

Other models, such as Moise+ [3], create an organizational middleware that checks whether actions of agents are allowed or not according to the governing organizational rules. The organization becomes an active entity that has the possibility to interfere in the agents decisions.

In our research we use human organizations as inspiration. From this point of view, we consider an organization as a description of roles, relations and interactions to

achieve certain coordination. The OperA approach fits well in that picture because of the expressive semantics and the distinction between organizational description and agent design.

2.2 Autonomous Agents in Organizations

We assume that the agents fulfilling organizational roles are autonomous entities; they have control over their internal state and their behaviour [4]. This implies that the organizational model specifies behavioural guidelines for the agents to assure desired features such as task coordination or information flow. The agents should follow those guidelines, but they are not forced to do so by definition. Previous studies have suggested several ways to let agents take up organizational rules.

The middleware of Moise+ [3] checks whether choices of agents are compatible with the organizational specification. The agents make decision locally, but the middleware can overrule the decisions of the agents. Therefore the agents are not fully autonomous in executing their choices.

Other organizational models, such as [5], are based on formal semantics and transition rules. The possible states of the organization are described by state transitions. The states include internal knowledge of the agents. Because agents have control over their internal state, this is not appropriate for our aims.

OperA [1] specifies contracts that describe the behavioural guidelines of a specific role. The internals of the agents are separated from the organizational specification. The only requirement is that the agents need to be able to understand the contracts. The agents are still fully autonomous in making decisions.

The OperA approach is compatible with our notion of autonomous agents. Furthermore, it has expressive semantics to define organizations. Therefore, we have chosen to use OperA for the organizational specification. In order to use OperA, we still need a mechanism to describe the adoption of organizational contracts into an agent's reasoning model. In Sect. 4 we show how autonomous agents can adopt organizational rules into their reasoning. In the following section we will describe the OperA model in more detail.

2.3 The OperA Model

OperA [1] provides a formalized specification language for agent organizations. OperA describes an operational organization in three parts:

- The organizational model: roles, relations, interactions
- The social model: population of organization, linking agents to roles
- The interaction model: interactions given the organizational model and the agents

The organizational model contains the description of the roles, relations and interactions in the organization. It is constructed based on functional requirements of the organization. The social model and the interaction model are the link between the organizational description and the executing agents. Here the organizational rules are translated to contracts for the agents fulfilling the roles. OperA includes a formal language to describe those contracts.

In an operational organization the social model and the interaction model can be dynamic, because of agents entering or leaving the organization. The organizational model is in principal static as long as no structural changes are carried through. The administrative tasks to keep track of the different organizational models are specified in organizational roles.

Agents enacting roles in a organization are expected to have some minimal knowledge about the concepts that are used to set up social contracts. The contracts are described in deontic expressions. The agents need to know the deontic concepts *permission*, *obligation* and *prohibition*. Furthermore, the description includes relations between roles. The agent needs to know the meaning of such a relation. For example, a *hierarchical* relation between role *r1* and *r2* implies that a request from *r1* is interpreted as an obligation by *r2*. OperA presents a formal description of the relevant concepts.

3 Example of an Organizational Description

In this section we present an example of an organizational model in OperA. We use a fire brigade to illustrate how an organization is specified and how the behaviour rules for the agents are constructed.

The fire brigade operates in a world where fires appear that need to be extinguished. In the organization we define two roles; *coordinator* and *firefighter*. The coordinator makes a global plan and tells the firefighters which fire they should extinguish. The coordinator has a global view of the world. The firefighters perform the actual tasks in the environment; they move to a fire location and extinguish the fires. They have only local views.

There is a hierarchical relation between the two roles, the coordinator is the superior of the firefighters and can send orders to the firefighters, which fire they have to extinguish. We want to show different forms of coordination within this organization. In our implementation, we achieve this by changing the autonomy level of the decision-making process of the firefighters.

A generic methodology to analyze a given domain and determine the type and structure of an application domain resulting in a OperA agent organization model is

Table 1. Methodology for designing agent organizations

	Step	Description	Result
OM	Coordination Level	Identifies organization's main characteristics: purpose, relation forms	Stakeholders, facilitation roles, coordination requirements
	Environment Level	Analysis of expected external behavior of system	Operational roles, use cases, normative requirements
	Behavior Level	Design of internal behavior of system	Role structure, interaction structure, norms, roles, scripts
SM	Population Level	Design of enactment negotiation protocols	Agent admission scripts, role enactment contracts
IM	Interaction Level	Design of interaction negotiation protocols	Scene script protocols, interaction contracts

described in [6]. The methodology provides generic facilitation and interaction frameworks for agent societies that implement the functionality derived from the coordination model applicable to the problem domain. Standard organization types such as market, hierarchy and network, can be used as starting point for development and can be extended where needed and determine the basic norms and facilitation roles necessary for the society. A brief summary of the methodology is given in table 1.

We focus on the organizational model or our firefighter organization. Below we define the coordination level, environment level and behaviour level.

3.1 Coordination Level

At the coordination level, the coordination type of the society is determined. There are several possibilities, for example a hierarchical model, a market based model or a network model. We have chosen for a hierarchical organization, since this is the most common structure in crisis management organizations such as our group of firefighters. The following characteristics are typical for a hierarchical organization:

- The leading goals for agents are global, organizational goals
- Relations between agents are fixed
- Communication is specified by design

Based on the choice for a hierarchical model we define the environment level and behaviour level.

3.2 Environmental Level

In the environment level, interaction between the organization and the environment is analyzed. Ontologies are needed to define organizational concepts and to define communication. Furthermore, the functional requirements of the organization are specified. This includes the global organizational purpose and the local objectives of the roles. We define coordination rules in terms of norms.

Organizational function. The purpose of our firefighter organization is to detect fires in the environment and extinguish them as soon as possible.

Ontologies. Besides OperA concepts to specify the organizational model, we need a communication language between the agents. We will use four performatives for the communication between the agents: *Request, Accept, Reject, Inform*.

Secondly we need a domain-level ontology to describe all objects in the environment, and the actions that the agents can communicate and reason about. Our domain ontology consists of one object, *Fire*, and one action, *Extinguish* and three states that describe the status of an agent with respect to its tasks: *Busy, Done, Free*. Furthermore the agents can send messages.

Roles. We do not consider external stakeholders of our organization. The roles in our organization are based on a functional analysis of their tasks. The roles are described in Table 2.

Table 2. Role table of the firefighter organization

Role	Relation to Society	Role Objectives	Role Dependencies
Applicant	Potential Member	Join Organization	Root
Root	From hierarchy model	Assign role to applicant	Applicant
Coordinator	From hierarchy model	Monitor Fires	
		Assign firefighter role	Applicant
		Monitor status of firefighters	Firefighter
		Assign fires to firefighters	Firefighters
Firefighter	Realization of extinguishing fires	Extinguish fires	
		Inform about status	Coordinator
		Announce assistance	Coordinator

A hierarchical organization needs a *root* role to take care of delegation of roles. The root role will give the role definitions to the agent highest in hierarchy and provide it with a social contract to specify the required behaviour. In our case, the highest role is the coordinator role.

The coordinator role has as objective to hire firefighters. It will assign the firefighter role to applicant agents. Furthermore it has the objectives to monitor the fires in the world, to monitor the firefighters, and to assign fires to firefighters. For the last two objectives the coordinator is dependent on the firefighters. We specify the firefighter role with the objectives to extinguish fires, to inform the coordinator about its status.

Dependencies between the roles appear from the description of the role objectives. We have defined the coordinator as the highest role in the organization. Therefore it has a hierarchical relation with the firefighter role, where the coordinator is the superior of the firefighters.

Table 3. The norms of the firefighter organization

Norm	
Situation	Handling extinguish-request
Responsibilities	Initiator: coordinator Action: firefighter
Triggers	Pre: coordinator sends extinguish-request Post: coordinator is informed about accept
Norm Specification	**Whenever** extinguish-request from coordinator **then** firefighter **is** obliged **to do** accept-request

Norm	
Situation	Announce status
Responsibilities	Initiator: firefighter Action: firefighter
Triggers	Pre: status change Post: inform coordinator about status
Norm Specification	**Whenever** status-change **then** firefighter **is** obliged **to do** inform-coordinator-about-status

Norms. We specify the norms that hold between the coordinator and the firefighter roles in table 3. In the first we describe how a firefighter agent handles requests to extinguish fires. The firefighter is obliged to accept the request from the coordinator. This norm follows directly from the hierarchical relation between the two roles.

We define a second norm telling that the firefighter should keep the coordinator informed about its status. The information is needed by the coordinator in order to do its tasks properly. This norm guarantees the required information sharing within our organization.

3.3 Behaviour Level

Here, we describe the social model and the interaction model as defined in OperA. Typically, a hierarchical organization has a relatively detailed social model and interaction model. This implies that the norms in the social model and the communication protocols in the interaction model do not leave much space for individual contracts with the agents. We assume that the behaviour rules as described in this level of the development of the organization match with the contracts with the agents.

Social Model. In the social model we define the social contract for the agents that fulfill the roles. The coordinator role just specifies the role objectives, with no further obligations. For the firefighter role we have defined some additional norms:

(1) Whenever extinguish-request from coordinator then firefighter is obliged to do accept-request
(2) Whenever status-change then firefighter is obliged to do inform-coordinator-about-status

As we explained, the first norm follows directly from the hierarchical relation between the two roles. Therefore, the social contracts for firefighter agents should only capture the second norm.

Interaction Model. The interaction model describes interaction contracts. An interaction contract between two agents describes a protocol that is followed by the agents during interaction scenes. Therefore all agents have interaction contracts for all interaction scenes.

There should be interaction scenes and contracts between applicant and root and between applicant and coordinator roles. However, they are only relevant to set up the organization. Therefore we just specify the interaction contracts between the coordinator and firefighter roles. This interaction scene will occur during the execution.

Interaction contracts are agreed upon by agents playing the roles and encountering interaction scenes. We assume the following interaction contracts between agents playing firefighter and coordinator, Table 4.

The first contract specifies the interaction between the firefighter and coordinator for the situation where the coordinator sends a request to the firefighter to extinguish a certain fire. The contract specifies that the firefighter is obliged to answer whether it accepts or rejects the request. The second contract specifies the interaction for the scenes in which the firefighter informs the coordinator about its status. The agents agreed that the coordinator does not need to respond.

Table 4. The interaction contracts of the firefighter organization

Interaction Contract	
Parties	Coordinator C, Firefighter F
Scene	Extinguish request
Clauses	**If** received(F, C, extinguish-request(fire)) **then** F **is** obliged **to do** answer(F, C, accept-refuse)

Interaction Contract	
Parties	Coordinator C, Firefighter F
Scene	Announce Status
Clauses	**If** received(C, F, status-report) **then** nothing

The interaction contract also poses a behaviour rule on the agent:

(3) Whenever extinguish-request from coordinator then firefighter is obliged to do answer-accept/refuse

3.4 Towards an Operational Organization

We have specified the organizational model in terms of roles, relations and interaction and we have defined contracts for agents that want to participate in the organization. Although agents are autonomous entities we expect them to follow the organizational rules. Therefore the agents should adopt the contract into their reasoning process. In the next section we describe how such a contract can be adopted by an agent using the reasoning model we described in [7].

4 Agents in Organizations

In Sect. 2, we have discussed other research concerning coordination of autonomous agents. The organizational middleware of Moise+ [3] can overrule choices of the agents, and therefore the agents are not fully autonomous in their choices. The organizational models based on formal transition rules [5] does not meet our requirement that the agent's internals have to be defined separately from the organization.

In this section we explain how autonomous agents can adopt organizational rules into their reasoning model. We believe that a modular approach in the agent's reasoning model is a promising way to adopt organizational rules into agent decision-making. By separating the organizational rules from the decision-making process of the agent we can guarantee the autonomy of the agent.

An autonomous agent should make its own decisions, and it decides how other agents can influence its decision-making process. In this section we introduce a component in the agent's reasoning model to deal with external events. We show how organizational rules can be adopted in this reasoning component.

4.1 Event Processing in the Agent Reasoning-Model

Here we explain briefly the reasoning model we described in a earlier paper [7] and we show how it can be used to adopt organizational rules into the reasoning process. In the

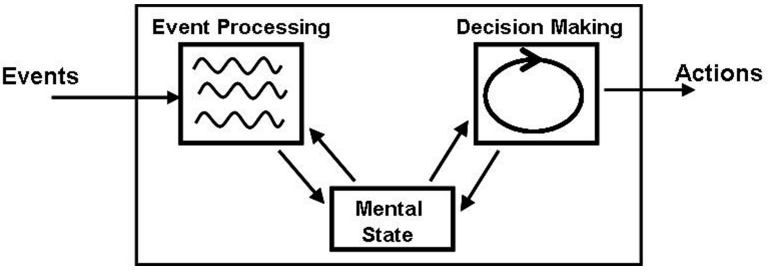


Fig. 1. Two phases in the agent's reasoning model

agent's reasoning-process we distinguish a phase for event processing and a phase for decision making as shown in Fig. 1. The event-processing phase gives the agent control over how it is being influenced by external events. The decide phase focuses on the decision of action.

In the event-processing phase the agent prepares the decision-making phase. External influences are processed here. External influence can be an agent's observations or messages from other agents. We have chosen to implement the event-processing phase with reasoning rules of the format same format as reasoning rules in BDI implementations such as 3APL [8] and AgentSpeak(L) [9]:

```
<HEAD> <- <GUARD> | <BODY>
```

The head of a rule is the event that triggers the rule. The guard should match the beliefs of the agent. The body of the rule expresses the influence of the event on the agent's reasoning process. For example, the following message describes that a request of a superior is to be accepted:

```
message(SENDER, request, TASK) <- superior(SENDER) |
      AddGoal(TASK)
```

The message is the trigger of the rule and the guard verifies with the agent's belief base whether the sender is a superior of the agent. If so, the request is accepted by adding the task to the agent's goals.

4.2 From Organizational Rules to Event-Processing Rules

We use the rules for event processing to specify how an agent's internal state is influenced by external events. The organizational specification describes behaviour rules that are meant to guide the agent's decision-making process. We propose to translate the organizational rules from the organizational description to event-processing rules for agent decision-making.

We have to define the set of required elements to translate organizational rules to event-processing rules. The way norms are specifies in OperA shows that norms are based on triggers. The triggers, which can be messages or observations, are external events, and therefore can be used as head of the event-processing rules. In the following, we propose to use a general message format consisting of a sender, a performative and the actual content. Observations must have the same format as agent beliefs.

Table 5. An ontology of event-processing rules that describe organizational rules

Rule element	Description	Possible values
Head	External event that triggers the rule	- message(Sender, Performative, Content) - observation(Content)
Guard	Situational constraints	Any belief set of the agent
Body	Effect on agent's mental state	- AddGoal(Goal) - AddBelief(Belief) - IgnoreEvent( )

The guards of the event-processing rules are restrictions based on internal beliefs of the agent. A guard can contain any set of beliefs. The example event-processing rule of the previous paragraph shows an example.

The body contains the effect of the external event on the internal state of the agent. This is of course dependent on how the internals of an agent are represented. We use the example of BDI agents. The effect of external events can then be described in term of belief changes and goal changes of the agent.

We describe behavioural rules as event-processing rules that result in a change of the agent's mental state. In Table 5, we give an example of how to translate organizational rules into event-processing rules. We show the values of the elements of the event-processing rules. We assume that we can translate all organizational rules into event-processing rules of the agent's reasoning process.

4.3 Prior Organizational Knowledge

In Sect. 2.3 we explained that some minimal prior knowledge is required. An agent taking up a role in the organization should know the meaning of deontic concepts and of relational terms. The meaning of the deontic concepts *obligation*, *permission* and *prohibition* are part of the ontology for event-processing rules. We translate an obligation for the agent using the *AddGoal* predicate. This predicate adds the task directly to the goal base of the agent.

We propose to translate permissions and prohibitions with the *AddBelief* predicate to describe that the task is permitted or prohibited. One could choose to define a predicate *RemoveGoal* to remove a prohibited goal. We prefer the addition of a belief that describes the prohibition, because removed information can not be recovered when the situation changes (e.g. when a goal is permitted again later on).

The agent should know the meaning of relational concepts, such as the hierarchical relation that we use in our example organization. The meaning of a hierarchical relation between an agent and its superior can be described by the following behaviour rule:

– Whenever an agent receives a request from a superior then the agent is obliged to accept the request

When we translate this behavioural rule to an event-processing rule using the above described elements we get:

```
(r1) message(SENDER, request, TASK) <- superior(SENDER) |
         AddGoal(TASK)
```

We gave this rule as example of an event-processing rule in Sect. 4.1. The above rule is considered to be general knowledge of the agent. The rule does not belong to a specific organization or a specific role.

5 Adopting Organizational Rules

Taking up a role in an organization means that an agent is expected to act following the constraints described in the role specification. In this section we show how organizational rules can be translated to event-processing rules for the agents. In the previous section we have shown how the required organizational prior knowledge is captured by the language specification and by some event-processing rules.

We continue with specific organizational rules using the example of the fire brigade. The firefighter organization shows three behavioural rules that a firefighter agent has to follow in that role. The rules are described by the norms in the social contracts and in the interaction contracts. The behavioural rules as described in Sect. 3.3 are:

1. Whenever extinguish-request from coordinator then firefighter is obliged to do accept-request
2. Whenever status-change then firefighter is obliged to do inform-coordinator-about-status
3. Whenever extinguish-request from coordinator then firefighter is obliged to do answer-accept/refuse

The first rule directly follows from the semantic meaning of the hierarchical relation between coordinator and firefighter. Therefore it is not part of the social contract. The other two rules are organization-specific and need to be described explicitly. As discussed in Sect. 4.2 the rules are triggered by events. They hold when certain conditions are true, and they result in expected reaction of the agents. We can translate those rules directly to reasoning rules for event processing using the language elements presented in 4.2.

```
(r2) observe( status-change ) <- TRUE |
       AddGoal( send(coordinator, inform, new-status) )

(r3) message( coordinator, request, extinguish(F) ) <- TRUE |
       AddGoal(send-answer(coordinator, request, extinguish(F)))
```

Organizational rules are part of the social contracts and interaction contracts that the agent agrees upon when its joins an organization. They can directly be transferred to event-processing rules. These reasoning rules capture all behavioural rules that belong to the role which the agent has taken up. We show that adopting those reasoning rules are a way make the agent aware of the organizational constraints.

The agent adds those reasoning rules to the event-processing phase that we have defined previously. This phase determines the autonomy of the agent's reasoning process; it determines the degree of external influence an agent allows into its reasoning. The agent limits the autonomy level of the decision-making phase with the organizational constraints.

6 Guarantee of Autonomy

The agent adds the event-processing rules derived from the organizational contracts to its own event-processing rules. We assume that, when an autonomous agent agrees with a contract, it deliberately chooses to do so. We further assume, that the rules for event-processing are possible and correct representations of organizational norms, so if the agent follows the event-processing rules it automatically follows the organizational norms.

In our reasoning model we have separated the event-processing rules from the actual decision-making process. This modularity has the advantage that it can reason about those rules. The agent knows the origin of the event-processing rules. Because the agent can make this distinction, it has the possibility to prioritize the event-processing rules and therewith it deliberately chooses to follow specific norms.

When organizational rules are embedded in the actual decision-making process, the agent will follow the norms implicitly. It might not be aware anymore of which norms it follows and it might not be aware of which norm belongs to which organization or role. Another advantage of the modular approach is that it becomes easy to add or change the event-processing rules, and thus change organizational norms.

Our approach allows for meta-reasoning about the event-processing rules. We claim that this guarantees autonomy of the agent; it knows how to follow the organizational rules, but it still has the possibility reason about them and take the chance to violate organizational norms. In the next section we give examples of meta-reasoning.

7 Bottom-Up Dynamics in Organizations

All static coordination mechanisms have their advantages and drawbacks. In a dynamic situation it is not possible to choose one coordination type that will always lead to the best performance. The main reason is that unexpected situations can occur that were not known at design time and that may not fare well with the selected coordination mechanism.

We have described a mechanism based on organizational concepts that specifies the coordination rules at an abstract level. At the same time, it preserves the actors' autonomy. However, the specified interaction rules in organizational models are static. When we follow the adoption of organizational rules from the previous sections, there are two ways to achieve dynamics in an organizational model:

- Top-down: a new organizational model is defined, and the agents change their contracts with the organization. As a consequence they adopt different reasoning rules for influence control, which will change the coordination.
- Bottom-up: the agents change the priority of reasoning rules for influence control by themselves if they notice that the organizational model fails. They adjust their autonomy to repair the organizational failure.

The top-down dynamics can be achieved by carrying out structural changes. Specific roles are required to start reorganizations. Bottom-up dynamics originate in autonomous choices of the agents. Our interest is in the bottom-up dynamics of the organization.

[7] showed dynamic coordination by allowing the agents choose their autonomy level. The agents acted following organizational rules, but also decided not to follow the rules in specific situations. Organizations in complex environments can benefit from adjustable autonomy of agents. In this section we argue how meta-reasoning about event-processing rules can achieve bottom-up dynamics in the organization.

Furthermore, an organization can benefit from the pro-activeness of agents and from the 'informal' actions and communication. Every organization is created for an objective. Its specification is meant to guarantee certain requirements, for example, about the information flow. The agents are often able to do many things outside of organizational specifications without violating any organizational rule.

Both issues, meta-reasoning and informal processes, are directly related to the autonomy of the agents. We discuss them in more detail and show the dynamics in organizations using example scenarios.

7.1 Meta-reasoning about Event-Processing Rules

An autonomous agent controls its internal state. Therefore, it should have control over external influences. The modular feature of our approach allows that, as it can reason about the event handling. The event-processing rules are derived from a contract with an organization. They can be role specific. At the same time an agent can have event-processing rules from other roles, or from itself.

We can tag the event-processing rules with their origin. For example, rule r1 from Sect. 4.3 is part of the agent's knowledge. It is not organization- or role-specific. Rules r2 and r3 from Sect. 5 are adopted via a contract with the firefighter organization. The agent can distinguish between different rules based on their origin. It knows that if it gives full priority to the organizational rules, it follows the organizational norms. It still has the possibility to prefer its own event-processing rules; however, this may lead to violation of organizational norms.

In [7] situations are described where organizations benefit from violation of norms. An agent's local observations can conflict with organizational rules. For example, if a firefighter is in danger he could ignore a request from the coordinator and give priority to his own goals in order to stay safe. He deliberately gives priority to his own goals, and therewith risks to violate the organizational norm.

The ability to distinguish between the reasoning rules can be used for prioritization of the rules. This can be done by applying machine-learning techniques. A learning agent can learn the situations in which specific rules should have priority, such as the *danger*-example.

Another option is to prioritize based on heuristics. Prioritization is studied in argumentation logics [10]. Argumentation as been applied to reason about interaction between agents [11] as well as to reason about norms [12]. Our rule-based approach of event-processing fits very well with this type of meta-reasoning.

Situation-based prioritization is an example of meta-reasoning. Given that an agent can reason about norms, it can use meta-reasoning to take different attitudes with respect to the organization. For example, an agent can adopt the organizational goals and drop its private goals, or it can still prefer its own goals above the organizational ones.

[13] and [14] describe several possible attitudes. The model presented in this paper allows those different types of agents.

7.2 Informal Processes

Meta-reasoning about event-processing rules guarantees agent autonomy and allows for norm violation. We argued that participant's initiatives besides the organizational guidelines structure are another benefit for the organization that follows from agent autonomy.

An organization is always specified for a certain purpose, possibly conflicting with the agents' individual purposes. Furthermore, the organizational rules guarantee required features, such as information flow, in order to optimize its performance. However, the agents are free to do what they want beside the organizational guidelines. For example, interaction protocols are defined to guarantee an certain distribution of information, but the agents can chat with each other and exchange knowledge without violating the norms. These informal processes are especially interesting when unexpected events occur that affect the organizational coordination mechanism.

We give the following example of a scenario of the firefighter organization that demonstrates the use of informal communication between firefighter clearly helps the organizational performance. The organizational rules specify the communication between the coordinator and the firefighters. Nothing is said about mutual communication between the firefighters. This implies that it is not forbidden to communicate. Therefore, we consider communication between the firefighters as informal communication. If two firefighters share their knowledge about a fire while extinguishing it, the extinguishing process might go faster. As a consequence, the organization performs better due to the informal communication.

Unexpected events that undermine the coordination can be overcome by informal processes. If, for example, the communication between some of the firefighters and the coordinator falls out, the organizational specification fails. The information flow as defined by the interaction protocols does not lead to the optimal knowledge for the coordinator; he misses the status of some of the firefighters. The informal communication between firefighters can be used by an individual firefighter to restore the information flow. If one firefighter tells another firefighter about its status, and this firefighter communicates it to the coordinator, the information flow in the organization is restored.

8 Conclusion and Future Research

Agents in an organization need to coordinate their actions in order to reach the organizational goals. Organizational models specify the desired behaviour in terms of roles, relations, norms and interactions. However, the actors in an organization are autonomous entities that control their internal state and their behaviour. In this research we have shown how organizational rules can be adopted by autonomous agents. We have developed a way to translate norms into reasoning rules for event-processing. The reasoning rule contains a trigger, situational constraints and an effect on the agent's mental state that are derived from the norm specification.

We have proposed a modular reasoning model to make organizational rules explicit. Since the agents are autonomous entities they will have their own reasoning rules next to the organizational rules. The modular approach makes that the agents can distinguish between different event-processing rules and that they are aware of the organizational rules. This allows for meta-reasoning about event processing, and gives the agent control over its internal state. It guarantees the autonomy of the agent and at the same time makes group coordination possible.

An important aspect of our method is the translation from the norms to reasoning rules of the agent. We have used a simple example of a firefighter organization to illustrate our ideas. More complex organizations might introduce complex behavioural rules and we have to evaluate whether we can express them in our language for event-processing rules. Furthermore, we argued that the possible effects on an agent's mental state are dependent on its decision-making process. This has implications for the translation as well.

We have presented some advantages of using modularity in the reasoning model. In our approach, the agent can do meta-reasoning reason about the event-processing rules, and thus about norms. We have used *prioritization* of event-processing rules as example of meta-reasoning. As future work we want to investigate different methods of meta-reasoning, such as argumentation-based techniques.

Acknowledgement. The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

References

1. Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. Utrecht University, PhD Thesis (2004)
2. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: ICMAS 1998: Proceedings of the 3rd International Conference on Multi Agent Systems, p. 128. IEEE Computer Society, Washington (1998)
3. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: SBIA, pp. 118–128 (2002)
4. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence 117(2), 277–296 (2000)
5. Matson, E., DeLoach, S.: Formal transition in agent organizations. In: IEEE International Conference on Knowledge Intensive Multiagent Systems (KIMAS 2005), pp. 235–240 (2005)
6. Dignum, V., Dignum, F., Meyer, J.J.C.: An agent-mediated approach to the support of knowledge sharing in organizations. Knowledge Engineering Review 19(2), 147–174 (2004)
7. van der Vecht, B., Dignum, F., Meyer, J.J.C., Neef, M.: A dynamic coordination mechanism using adjustable autonomy. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 83–96. Springer, Heidelberg (2008)
8. Dastani, M., van Riemsdijk, B., Dignum, F., Meyer, J.J.C.: A programming language for cognitive agents: Goal directed 3apl. In: Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) PROMAS 2003. LNCS, vol. 3067, pp. 111–130. Springer, Heidelberg (2004)

9. Rao, A.S.: Agentspeak(l): Bdi agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS (LNAI), vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
10. Brewka, G.: Reasoning about priorities in default logic. In: AAAI 1994: Proceedings of the twelfth national conference on Artificial intelligence, vol. 2, pp. 940–945. American Association for Artificial Intelligence (1994)
11. Kakas, A., Maudet, N., Moraitis, P.: Modular representation of agent interaction rules through argumentation. Journal of Autonomous Agents and Multi-Agent Systems 11(2), 189–206 (2005)
12. Oren, N., Luck, M., Miles, S., Norman, T.J.: An argumentation inspired heuristic for resolving normative conflict. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428. Springer, Heidelberg (2008)
13. Sichman, J., Conte, R.: On personal and role mental attitude: A preliminary dependency-based analysis. In: de Oliveira, F.M. (ed.) SBIA 1998. LNCS (LNAI), vol. 1515, pp. 1–10. Springer, Heidelberg (1998)
14. Dastani, M., Dignum, V., Dignum, F.: Role assignment in open agent societies. In: AAMAS 2003, pp. 489–496. ACM Press, New York (2003)

Combining Job and Team Selection Heuristics

Chris L.D. Jones and K. Suzanne Barber

The University of Texas at Austin
Laboratory for Intelligent Processes and Systems
1 University Station C5000, Austin, TX, 78712-0240
{coldjones,barber}@lips.utexas.edu

Abstract. In open markets and within business and government organizations, selfish agents often face the question of what tasks to work on, and what partners to work with. Optimal solutions are particularly difficult to find in large-scale, unpredictably dynamic environments. Previous work has examined the use of separate job selection and team selection heuristics to guide agent decisions in these domains, but did not explore how these decisions influence each other, and how these heuristics should therefore be used together. Accordingly, this paper presents a mechanism for combining job and team selection heuristics for agents operating in a large-scale, unpredictably dynamic environment. An experimental analysis of this new job/team selection mechanism demonstrates significant improvements in agent performance, both in terms of credit earned and in percentage of jobs successfully completed.

Keywords: Coalition formation, Task selection, Partner selection, Fault-tolerance, Dynamic environments, Large-scale environments, Request for Proposal.

1 Introduction

The Request For Proposal (RFP) domain is a widely applicable model wherein problems are divided into a set of required subtasks, which are in turn worked on by a team of agents. RFPs are often found in business-to-business environments, where specialized expertise providers can focus on different aspects of complex problems in a market environment. Furthermore, the combination of RFPs with peer-to-peer level services such as the Amazon Mechanical Turk [2] suggest markets on a far larger scale, where both businesses and individuals submit RFPs for almost every type of problem imaginable.

Work by Kraus et al. in the RFP domain focused on static problems, where a problem's required subtasks did not change once the problem was submitted [12]. However, incomplete information and bounded rationality may contribute to a faulty initial perception of a problem's requirements. Real world problems also involve inherent environmental dynamics that may change a problem's requirements over time. Furthermore, fully autonomous agents in large-scale, open environments such as the Internet may operate without enforceable contracts, meaning that sudden defections by team members are possible.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 33–47, 2009.

Accordingly, recent work in the field has focused on dynamic RFP environments, where subtasks are added and subtracted from the set of problem requirements as the problem is being worked on [9]. Specifically, algorithms have been proposed to select problems (jobs), teams, and task assignments so that sudden changes, such as added task requirements or team defections, could be adapted to. For example, by maximizing the number of auxiliary skills found in a team, agents on the team may better cope with the addition of new subtasks.

However, this prior work did not adequately explore how job selection and team selection decisions influence each other. For example, based purely on expected earnings an agent might greedily select a job requiring skills that no potential partners possess. Alternatively, a job that may initially appear unattractive may be decomposable in a way that fits perfectly with the skill set of available partners.

This paper therefore presents a mechanism for combining job selection and team selection heuristics for agents operating in a large-scale, unpredictably dynamic environment. Particularly, the paper conducts an experimental analysis that compares an earlier mechanism found in [9], where job selection and team selection heuristics were used separately, to a new protocol where job and team selection heuristics are used in combination. The analysis is conducted in the context of a large-scale agent simulation (1000 agents) and examines multiple metrics of agent performance across a range of agent heuristics and levels of environmental dynamicism.

This paper is organized as follows. In section two, it discusses related work in the multi-agent systems community. In section three, it presents a new mechanism for combining job and team selection heuristics, and compares this mechanism to an earlier mechanism for selecting jobs and teams separately. In section four, it describes the setup and parameters of a simulation to test the relative utility of the proposed selection mechanisms. In section five it discloses the results of the simulation, and analyzes those results. Finally, section six suggests several ideas to expand upon those strategies.

2 Related Work

Coalition formation has been studied both inside and outside of the multi-agent systems community for some time. Some research has focused on the formation of optimal coalitions by a centralized authority, [19] while other research has focused on the formation of coalitions to solve jobs by a hierarchical structure of agents [1], or the formation of institutions through assumption-based argumentation mechanisms [5]. Still further research has been focused coalition formation between selfless agents in a dynamic [15] or open environment, [20] or between agents willing to delegate their autonomy to a centralized controller or consensus decisions among groups of agents [14].

However, such research has limited applicability to decentralized selfish agents, which may be unwilling or unable to take direction from a centralized authority. Other work has therefore examined selfish agents operating in various environments. Research has focused on building coalitions of agents who lack a common view of a coalition's value, [7] as well as coalitions developed between rationally-bounded agents, [18] or agents who lack full knowledge about the abilities of potential partner agents [19]. Others have examined the interaction between agents in market environments [23]. Such research frequently focuses on relatively small groups of agents,

although still other research has focused on the use of congregations, [3] adaptive social networks, [6] and even physics-motivated mechanisms to allow large groups of agents to form large, mutually beneficial coalitions [13]. It should be noted, however, that such work is frequently focused on only one possible task at a time, or does not occur in dynamic, unpredictable environments.

In contrast, Klusch and Gerber focus on the formation of coalitions of agents to work on multiple possible tasks in dynamic environments, by utilizing a simulation-based scheme to determine the utility of various potential coalitions in a given state of a dynamic environment [11]. This work allows for complex negotiations between the different potential partners of a coalition, and takes risk vs. reward considerations into account when considering different potential coalitions. However, it differs substantially from the work below in that the coalitions formed are not adaptive once formed, nor are jobs selected based on the potential teams available to solve a given job.

Oren et al. have recently proposed an argumentation-based mechanism for allowing agents to minimize the number of norms they violate as changing circum-stances demand [16]. Such work may be very useful in shaping long term agent behavior, as agents that follow norms are rewarded and those that violate norms are punished or ostracized, but norms have limited usefulness in improving the robustness of a one-shot team, since agents who defect from the team may never be encountered again, and thus face no penalty for their defection.

Tambe et al.'s work is likewise relevant, wherein selfish agents are collected into a team by an initial authority, often a human programmer. The agents may then be delegated by software algorithms into roles which pursue various sub-goals critical to the overall mission [22]. As will be shown in further detail below, the strategies described in this paper build on this research by allowing agents to form adaptive teams without the need for an initial organizing authority.

In addition, Soh and Tsatsoulis have focused on the possibility of hastily-formed coalitions in response to new problems or events [21]. This research forms the basis for one of our heuristics for job selection, as will be described in further detail below.

3 Job and Team Selection

This paper introduces a new mechanism for combining job selection and team selection heuristics, which operates using the same general experimental model and agent protocol as [9]. More specifically, section 3.1 below reviews the formal model for agents and jobs, while section 3.2 discusses a new mechanism named Combined Job/Team selection (CJT) for selecting potential jobs and teams, as well as how CJT compares to a previous job and team selection mechanism, referred to here as Separate Job/Team selection (SJT). Section 3.3 reviews specific job and team selection heuristics used by both the old and new selection mechanisms.

3.1 Agents and Jobs

Consider a set of general tasks $T = \{T_i\}$, where $1 \leq i \leq \alpha$. Each general task T_i represents a type of job that an agent might carry out: if T is limited to tasks involved in building construction, for example, T_1 might be building a driveway, while T_2 might

be constructing a roof, and so on. Each general task T_i is therefore a set of task instances $\{T_{ij}\}$, where each T_{ij} is a specific instance of general task T_i associated with a job J_j, and where each job J_j is part of a set of jobs $J = \{J_j\}$, where $1 \leq j \leq \beta$. For example, if T represents the set of all tasks associated with building a building, and if J is the set of all buildings under construction, then T_{11} might be building a driveway at a first building under construction, T_{12} might be building a driveway at a second building under construction, T_{21} is constructing a roof at the first building, and so on.

Each job J_j in set J contains a potential task instance T_{ij} of every possible task T_i in T, but only a subset of these tasks need to be completed to finish the job. Again, returning to the building example, every building in existence could conceivably have a swimming pool, or a loading dock, or a conference room, but in practice factories and offices rarely have swimming pools, and houses rarely have loading docks. Accordingly, within each job J_j, task instances T_{ij} are separated into a set of active task instances $ActiveTasks_j$, all of which must be completed for the job to be finished, and a set of inactive task instances $InactiveTasks_j$, which are irrelevant to the job's completion status. For any job J_j, $ActiveTasks_j \cup InactiveTasks_j = \{T_{ij}\}$ for all i, and $ActiveTasks_j \cap InactiveTasks_j = \varnothing$.

Skills, Credit and Dynamic Jobs. Continuing on, a set of skills $S = \{S_i\}$ and a set of self-interested agents $A = \{A_k\}$ are introduced, where once again $1 \leq i \leq \alpha$ and $1 \leq k \leq \chi$. Each skill S_i is associated with a general task T_i, and may be used to work on and eventually complete any task instance T_{ij} in T_i. Furthermore, each agent A_k has an associated set of skills $AgentSkills_k$ that A_k is capable of doing, where $AgentSkills_k$ is a subset of S. Each agent A_k has the same number of skills, and each skill in S equally common among agents in A.

Agents use associated skills to complete tasks associated with various jobs, and earn credit by completing all active tasks in a given job. More specifically, each task instance T_{ij} has an associated $TaskLength_{ij}$, where $1 \leq TaskLength_{ij} \leq \gamma$. To complete task instance T_{ij}, an agent A_k must use an appropriate skill S_i on the task instance for $TaskLength_{ij}$ timesteps. Accordingly, function $C(T_{ij})$ is defined as a value ranging from 0 to $TaskLength_{ij}$, and represents the amount of time that one or more agents have worked on T_{ij}. Different tasks are worth the same amount of credit, but agents earn rewards proportional to the $TaskLength_{ij}$ of any task instance T_{ij} they have finished. For example, an agent that completes a task over five timesteps earns five credit points, while an agent that completes a task over eight timesteps earns eight credit points.

To simulate the end results of uncertain information, bounded agent rationality and dynamic, unpredictable environments, jobs in J are dynamic and unpredictable. More particularly, task instances T_{ij} in J_i are randomly moved between $ActiveTasks_i$ and $InactiveTasks_i$ on a periodic basis. This may be best understood as a sudden change to a job's solution requirements. For example, despite the best efforts of project management and requirements engineering, software development projects frequently change their required functionality in the middle of development, making some already-completed portions of the project obsolete and requiring new modules to be built from scratch. Similarly, task instances T_{ij} which are moved from $InactiveTasks_j$ to $ActiveTasks_j$ must be done from scratch, while only active task instances which have been fully completed are immune from being moved to $InactiveTasks_j$. (Admittedly, it is not unheard-of for fully-completed portions of many different types of

projects to be discarded, but it is also reasonable to argue that work which has been fully completed is often used in some way, somehow, whereas partially completed work is often abandoned entirely.) Note that the number and types of tasks that must be completed for an individual job to be completed is therefore continually changing.

3.2 Selection Mechanism

Agents operating in the environment described above face two primary decisions: what jobs to work on, and what agents to partner with. More precisely, freelance agents fulfill one of two roles at any given moment, and make different decisions in each of these roles. In the first role, an agent takes a proactive "foreman" role and has the ability to select both an available job to work on, and a team of agents capable of solving the currently active tasks associated with that job. The foreman then attempts to assemble a selected team of agents to work on a selected job by sending proposal messages to one or more potential partners. Alternatively, in the second role, agents act in a reactive "worker" role and must decide whether to accept the proposal offers sent out by a foreman agent, or whether to continue with the current job they are working on, if any.

Agents in either role utilize paired job and team selection heuristics, described in further detail below, to compare and rank different jobs and potential teams. For example, a Greedy job selection heuristic might rank the potential profitability of different jobs based on how much credit an agent could earn by completing assigned tasks in a job, while a Redundant team selection heuristic might rank the robustness of different teams by how many redundant skills each team has to be utilized in case an agent defects from a team.

However, of greater interest in this paper is the mechanism by which heuristic information is utilized to select jobs and teams. More precisely, the newly proposed job and team selection mechanism CJT uses the product of a normalized job and a normalized team selection heuristic for almost every selection decision, as opposed to the earlier SJT mechanism from [9], which used only one heuristic at a time.

CJT Mechanism. Agents acting in the foreman role initially look at all jobs in J that are not currently being worked on by another team. (As a simplifying assumption, both jobs and agent skills are globally accessible data.) Under CJT, the foreman utilizes a job selection heuristic to select δ available, top-ranked jobs from J, thereby creating set $P = \{J_w\}$, where $1 \leq w \leq \delta$. For each J_w in P, each foreman agent then generates ε agent teams capable of solving J_w. More specifically, these agent teams each include the foreman agent which generated the team, and one or more other agents A_k, such that the combined skills of all the agents in the team are capable of completing the task instances in $ActiveTasks_w$ of associated job J_w. These teams thus form a set of teams $Q = \{Team_x\}$, where $1 \leq x \leq \delta\varepsilon$. Furthermore, each agent A_k in a $Team_x$ is associated with $AssignedInstances_{kw}$, which, after a team has been formed, represents the set of task instances T_{iw} that each agent A_k in the team is assigned to complete in job J_w. Teams are currently assembled via a semi-random approach that seeks to satisfy the various solution requirements one at a time, but nearly any constraint satisfaction solver could also be used.

Under the CJT mechanism, once Q is generated each foreman agent uses a selection heuristic R to rank each team in Q, where R is the product of a normalized job selection heuristic and team selection heuristic. R accordingly contains information about both the value of J_w to the foreman agent, via the job selection heuristic, and the potential robustness of $Team_x$, via the team selection heuristic.

Likewise, an agent acting in the worker role under CJT uses R to decide between incoming proposals and the job and team it is currently part of, if any. Note that, since teams must successfully adapt to sudden changes in *ActiveTasks* and team composition to earn credit, as will be described in further detail below, a job/team selection mechanism such as CJT that takes both potential job profit and team robustness into account is hypothesized to provide a significant improvement in credit earned by an agent.

SJT Mechanism. In comparison, SJT the selection mechanism described in [9] utilizes only one heuristic at a time. For example, like CJT, a foreman agent using SJT uses a job selection heuristic to rank top-ranked jobs from *J*. However, unlike CJT, SJT selects only the one or more jobs that "tie" for the top ranking, according to the job selection heuristic, meaning that size of the generated set P is at most δ, and often less. Set Q is again created by generating ε teams for each job in P. These teams are then ranked only by a team selection heuristic, and the foreman agent selects the top rank team to propose to other agents. Note that because each job in P is tied according to the job selection heuristic, the potential profitability of all jobs in Q is the same, and only the structure of the team distinguishes job/team pairings. However, because SJT does not consider slightly lower-ranked jobs, it may not consider job/team pairings where the lower ranking value of the job is offset by the increased adaptability of the team.

Furthermore, worker agents under SJT use only a job selection heuristic to decide between new proposals and their current job/team assignments. Agents using SJT may therefore overlook job/team pairings where increased team adaptability offsets lower potential job profit. In contrast, because CJT considers both job and team heuristics, a worker agent is likely to leave its current assignment only if the new job/team pairing shows a significant increase in R.

3.3 Selection Heuristics

As described above, agents utilize job and team selection heuristics to select jobs to work on and teams to work with. Each agent is associated with one job selection heuristic and one team selection heuristic, which are used by the selection mechanisms described above. More particularly, job and team selection heuristics are normalized, so that each heuristic produces a value between 0 and 1 which describes how desirable that job or team is, with 0 being completely undesirable and 1 being completely desirable. Job heuristics attempt to maximize agent profit by selecting attractive jobs, while team heuristics attempt to select teams that are structured to handle sudden changes in the environment or inside the team itself.

The first job selection heuristic is the pre-normalized Greedy heuristic (Eqn. 1) that maximizes the expected reward from a job J_j. While a naive heuristic would simply choose jobs that require the greatest amount of work (and thus the greatest amount of associated reward), the Greedy heuristic takes the dynamicism of the environment into account by giving double weight to portions of a task that have already been

completed, thereby giving preferential treatment to large jobs that are less likely to undergo changes before the job is complete. The heuristic is normalized by dividing an agent's potential profit by the number of skills per agent times the maximum length of each task, which represents the maximum theoretical profit an agent could earn from a job.

$$\max_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} + C(T_{ij})) \tag{1}$$

The second job selection heuristic is the normalized Lean heuristic (Eqn. 2) that minimizes the amount of work needed to complete a job, thereby letting agents opportunistically form teams to quickly solve simpler problems, similar to [21].

$$\frac{1}{\min_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} - C(T_{ij}))} \tag{2}$$

Note that these mechanisms stand in contrast to previous work in task selection under uncertain conditions, such as Hannah and Mouaddib [8], where a problem's uncertain elements are explicitly modeled probabilities. Instead, the heuristics described here operate under any level of uncertainty, from any source. However, future work is possible where the above heuristics are adaptive based on a known or suspected level of uncertainty in the environment, or in a specific problem.

The first team selection heuristic is a Null heuristic that does not rank the teams, but rather keeps teams ordered according to how the strategy's job selection heuristic ranked the jobs associated with each team. This effectively eliminates the team selection heuristic from both selection mechanisms.

The second team selection heuristic is the normalized Fast heuristic (Eqn. 3) that minimizes the maximum amount of work that any member of a $Team_x$ needs to complete. Alternatively, the Fast heuristic could be said to minimize the amount of time needed before the entire team has completed work on associated job J_w.

$$\frac{1}{\min_x \left[\max_{A_k \in Team_x} \sum_{T_{iw} \in AssignedInstances_{kw}} (TaskLength_{iw} - C(T_{iw})) \right]} \tag{3}$$

The third team selection heuristic is the normalized Redundant heuristic (Eqn. 4) that seeks to maximize the number of redundant skills in $Team_x$. In other words, the Redundant heuristic prefers teams with multiple agents capable of working on active task instances, thereby increasing the ability of a team to deal with the defection of an agent.

$$\max_x \frac{\sum_{T_{iw} \in ActiveTasks_w} \begin{cases} 1 < \sum_{A_k \in Team_x} |T_{iw} \cap AgentSkills_k|, & 1 \\ \text{Otherwise}, & 0 \end{cases}}{|ActiveTasks_w|} \tag{4}$$

The fourth team selection heuristic is the normalized Auxiliary heuristic that seeks to maximize the number of auxiliary skills in $Team_x$. In other words, the Auxiliary heuristic (Eqn. 5) tries to maximize the combined skills of a team that are not immediately applicable to task instances in $ActiveTasks_w$, thereby increasing the ability of the team to deal with newly added task instances.

$$\max_x \frac{\left| InactiveTasks_w \cap \bigcup_{A_k \in Team_x} AgentSkills_k \right|}{\left| InactiveTasks_w \right|} \tag{5}$$

Note that, intuitively, the Fast, Redundant, and Auxiliary heuristics each prefer a greater number of partners in a team, since this increases the amount of work that can be done in parallel and the number of unused skills for each partner. Alternatively, the normalized MinPartner heuristic (Eqn. 6) prefers teams with the smallest number of partners, thereby implicitly using a greater number of skills per partner and thus a greater amount of potential profit per partner.

$$\frac{1}{\min \left| Team_X \right|} \tag{6}$$

Each of these five team selection heuristics is combined with each of the job selection heuristics for experimental comparisons, as described in further detail below.

4 Experimental Setup

To evaluate the relative utility of CJT and SJT, both mechanisms are tested in a simulation environment wherein agents compete to form teams and solve jobs according to the described strategies. More particularly, the experimental setup described below was executed twice, once for CJT and once for SJT. Agent performance under each mechanism was evaluated under multiple metrics described below, with statistical methods used to detect significant differences.

Agents in set *A* were divided into ten different classes, each of which contains an identical number of agents, and each of which implements a different team formation strategy consisting of a paired job and team selection heuristic. By assigning agents credit for each task instance completed, the relative utility of each strategy may be determined by comparing the average amount of credit earned by each class of agents. Furthermore, by varying the rate of change of the solution requirements for different jobs ("dynamicism"), the relative performance of these strategies in a dynamic environment can be determined.

Agents in set *A* operate in a simulation environment that is divided into discrete timesteps, or rounds. During each round, each agent may coordinate with other agents in *A* to form teams, or, if it is part of a team, may work on a task instance associated with a specific job in *J*. Each agent in *A* can belong to, at most, one team at a time, and each team works on only one job at a time. This is arguably a simplistic assumption, since real world providers of valuable skills or expertise frequently multitask between different

projects at the same time. However, many problem solutions require complete focus from the workers involved, or security or other constraints may require exclusivity. Furthermore, requiring each agent to be part of only one team at a time allows us to clearly delineate where an agent is making a contribution. Determining to what degree an agent's partial efforts require task reassignments touches on complex multidimensional trust issues [7], and as such is too complex to be addressed here.

During each round, an agent A_k may work on a job J_j by utilizing a skill S_i found in $AgentSkills_k$ to work on a task instance T_{ij} found in set $ActiveTasks_j$. Each agent utilizes only one skill in any given round. After A_k has worked on T_{ij} for a given number of rounds, T_{ij} is completed.

Credit. Credit is distributed to agents when a job J_j is completed, which, in turn, occurs when all task instances in $ActiveTasks_j$ are completed. Upon job completion, credit points for each active, completed task are given to the agent which completed the task. As described above, these credit points are proportional to the length of the completed task (e.g. a completed task of length five would give five credit points). No credit is given for work on task instances that were moved to $InactiveTasks_j$ before completion, or to agents who worked on, but did not finish, a completed task instance. Accordingly, agents in the simulation may be said to work in a "pay-for-play" environment, where credit is distributed directly to those who have fully completed a given job.

Once a job J_i has been completed and paid out its credit, it is removed from J and a new J_j is created and inserted in J. Each new J_j starts with the same number of task instances randomly placed in $ActiveTasks_j$, and task instance in the new job must be completed from scratch.

As described above, the simulation incorporates unpredictability by shuffling task instances between $ActiveTasks_j$ and $InactiveTasks_j$. More particularly, each round a given percentage of jobs in J are randomly selected to be shuffled. This percentage is referred to as the dynamicism of the simulation. Each task instance in each selected job J_j has a random chance of being selected for shuffling between $ActiveTasks_j$ and $InactiveTasks_i$, such that, on average, one task instance per selected job is shuffled. However, as described above, task instances that have been fully completed cannot be moved from $ActiveTasks_j$.

4.1 Team Formation

As described above, teams are formed by a foreman agent. The opportunity to act as a foreman agent is randomly distributed among agents, such that any given round of the simulation a given percentage of agents will have the opportunity to form teams. Once the foreman agent has used its associated team formation strategy to select a potential team $Team_x$ and job J_w, the foreman claims the job so that no other foreman can work on it and sends proposal messages to potential partners in $Team_x$ indicating the $AssignedInstances_{kw}$ that a potential partner would work on. Note that, to encourage agents to form teams, $|AgentSkills_k|$ is constant for all k, and the initial value of $|ActiveTasks_i| > |AgentSkills_k|$.

When these proposal messages are received, each agent ranks the $AssignedInstances_{kw}$ it is currently working on (if any) against one or more proposed $AssignedInstances_{kw}$ using the SJT or CJT mechanisms described above. If the agent finds that its current assigned tasks are preferable to any of the proposals, it continues working on its current job, and the lack of a response is taken as a decline message by the foreman which sent

the proposal. If the agent receives a proposal it finds more attractive than its current job assignment, the agent returns an accept message to the foreman which sent the proposal.

Accordingly, it is noted that agents may stop work on their current assignments at any time upon receiving a more attractive proposal (or, if they become a foreman agents, upon finding a more attractive job to work on). This obviously runs counter to a significant amount of existing work in contract negotiation and breaking contracts, wherein contracts are unbreakable or explicit penalties are calculated beforehand. [4]. However, such work usually assumes some viable mechanism for contract enforcement. In contrast, the scheme described here better simulates many environments where contracts are largely nonexistent or unenforceable (i.e. informal task forces and many Internet transactions). This lack of commitment between agents, combined with the dynamicism and unpredict-ability of jobs within the simulation, also makes it desirable for agents to assemble teams that can survive agent defections and changes in the task instances required to finish the job.

If the foreman does not receive accept messages back from all potential partners within one time step, the team formation process has failed and the foreman, as well as the agents which accepted the team proposal, must wait for new team proposals or for their next chance to be a foreman. Claimed job J_w is freed so other agent can work on it.

Once agents have successfully formed $Team_x$, they begin to work on the task instances associated with J_w. Non-foreman agents may work on J_w until they have completed all task instances in $AssignedInstances_{kw}$. In contrast, the foreman agent may stay with J_w until the job is complete, even if the foreman has completed its assigned tasks. While J_w is incomplete, if a non-foreman agent defects from the job, or a new task instance is moved into $ActiveTasks_w$ set, the foreman is responsible for finding an agent to work on the new or abandoned task instance. The foreman may therefore assign the new task instance to the $AssignedInstances_{kw}$ set of itself or a partner agent, in a manner similar to the team reformation strategies in [22]. If no team member has the skill required to handle the new task, the team has failed and dissolves with the job uncompleted. A new team which later tries to claim the job must begin the job from scratch. It therefore follows that teams must handle defections and new task instances to be successful.

Experiments were conducted using the basic parameters in Table 1, which were selected to broadly model a moderately large problem-solving market. The experiments

Table 1. Experimental parameters

Parameter	Value
Number of classes	10
Agents per class	250
Per round chance of agent acting as foreman	1%
Jobs	1000
$\|T\|$	20
$\|AgentSkills\|$	5
Initial size of $\|ActiveTasks\|$	10
Range of TaskLength	1 to 10 rounds
Credit received per round of completed task instance	1
Number of potential teams examined per top-rank job	15
Dynamicism range	0% to 100%, 25% increment
Number of rounds per simulation	2500
Number of simulations per dynamicism step	20

tested all strategies against each other simultaneously to better understand their relative performance in a field of heterogeneous agents. Furthermore, agents were assigned randomly to each class with a flat distribution to ensure that the results were not specific to a precisely equal distribution of agents.

5 Results and Discussion

Figure 1 displays the credit earned by the various classes of agents as both foremen and workers as the level of dynamicism and the job/team selection mechanism are

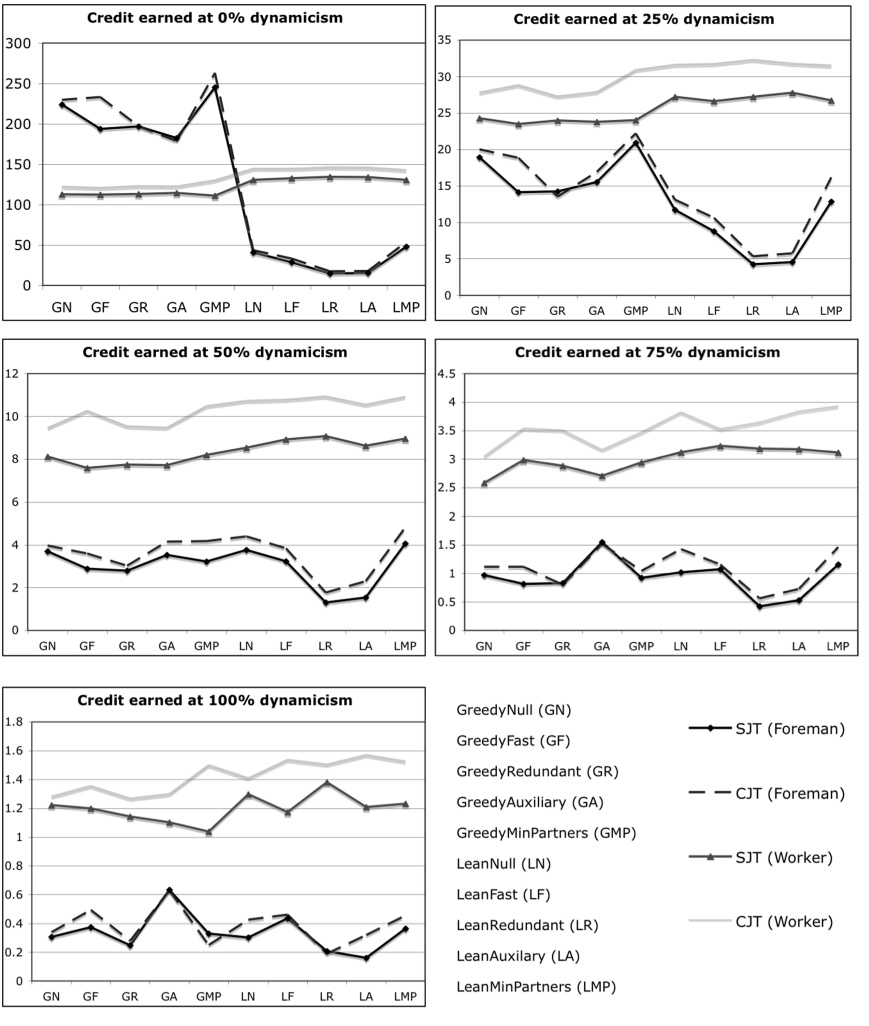


Fig. 1. Comparison of foreman and worker credit earned by dynamicism, strategy, and selection mechanism

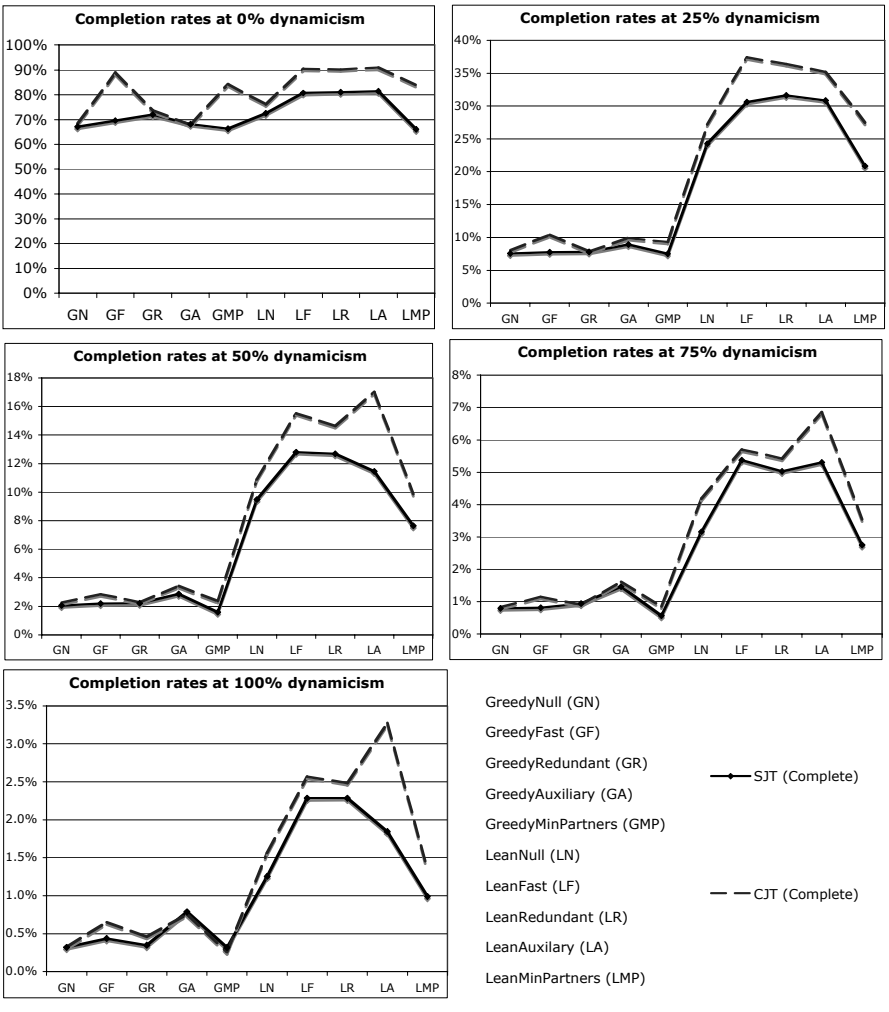


Fig. 2. Comparison of job completion rates by dynamicism, strategy, and selection mechanism

varied. As can be seen from the graphs, the credit earned by foremen agents utilizing the Combined Job and Team (CJT) selection mechanism equaled or exceed the credit earned by foremen agents utilizing Separate Job and Team (SJT) selection in most cases. More precisely, a pair-wise t-test comparison (alpha = .05 for all statistical comparisons) of credit earned by agents using the old and new job selection mechanisms shows that in all cases, the credit earned by the foremen agents using CJT significantly exceeded or statistically tied credit earned by agents utilizing SJT.

Furthermore, broadly speaking, agents utilizing the Lean job selection heuristic benefited more from the use of CJT than did agents utilizing the Greedy job selection heuristic; all Lean agent classes at 0%, 25% and 50% dynamicism earned significantly more with CJT than with SJT. This may be due to a "halo" effect that has less

to do with the foreman's decisions made using CJT, and more do to with the decisions of worker partner agents made using CJT, since worker agents would be less likely to leave a robust team for a job that could be completed more quickly.

A stronger trend can be seen in the earnings of worker agents utilizing CJT. As can be clearly seen from the graphs in Figure 1, all worker agents earned more with CJT. Furthermore, t-test comparisons indicate that all agent classes at all dynamicism levels earned significantly more using CJT than SJT. Again, this makes sense if we consider a potential halo effect, where the combined use of job and team selection heuristics may make agents less likely to immediately abandon one team for another, based purely on a difference in the job rankings.

More support for this idea may be found in Figure 2, which shows the job completion success rate of agent teams after formation. As with Figure 1, a clear trend can be seen wherein agent classes using CJT complete a greater or equal percentage of jobs than agent classes using SJT.

T-test comparisons between success rates for CJT and SJT show that, for all data points, agent classes using CJT either statistically tied or significantly improved on the success rate of agents using SJT. It is noted that, because the job completion success rate depends directly in part on the proclivity of partner agents to stay with a formed team, an improvement in the job success rate is strongly indicative that partner agents are less likely to leave a formed team when using a combination of job and team selection heuristics.

It is also noted that, while variations in the amount of credit earned by different agent classes may depend largely on various protocol assumptions (e.g. how much credit an agent earns for completing a task abandoned by another agent) the success rate of a formed team is a direct measure of how successful various agent classes are at forming teams which can successfully adapt to sudden changes job requirements and team formation. Furthermore, agents using CJT maintained a consistent advantage, even as dynamicism increased to very high levels (at 100% dynamicism, each job, on average, added or dropped one subtask per timestep).

Finally, statistical comparisons of the total number of jobs completed and credit earned by all agents at all dynamicism levels shows that agents utilizing CJT invariably completed significantly more jobs and earned significantly more credit than agents using SJT.

6 Conclusions and Future Work

This paper presented a novel mechanism for combining job selection and team selection heuristics for agents operating in a large-scale, unpredictably dynamic environment. This mechanism allowed agents to decide between potential job/team combinations when acting in a "foreman" role, and to better weigh new job offers against current assignments when acting in a "worker" role by utilizing the product of normalized job and team selection heuristics. In comparison, earlier selection mechanisms used job and team selection heuristics in isolation.

The paper then conducted an experimental comparison of different classes of agents utilizing the new job/team selection mechanism with agents using the old job/team selection mechanism while varying environmental dynamicism. Results

indicated that the new mechanism equaled or significantly improved the performance of agents in both the worker and foreman roles, both in terms of credit earned and in terms of percentage of jobs successfully completed.

This paper therefore proved that a job/team selection mechanism which takes both job and team selection heuristics into account when making decisions improves mechanisms which make these decisions in isolation. This new mechanism works for agents in a large-scale environment (>1000 agents) and allows agents to successfully complete jobs even in highly dynamic and unpredictable circumstances.

A number of possible avenues of investigation suggest themselves, the most obvious of which is to further investigate if other mechanisms for combining job and team selection heuristics would improve agent performance even further. For example, heuristics could be combined with either a static or dynamic weighting. Other potential work includes a deeper theoretical investigation of how specific job and team selections interact with each other. Finally, the experimental environment could be broadened to give different tasks different reward values, or take different and/or changeable quality of service from different agents into account.

Acknowledgments. This research is sponsored in part by the Defense Advanced Research Project Agency (DARPA) Taskable Agent Software Kit (TASK) program, F30602-00-2-0588. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] Abdallah, S., Lesser, V.: Organization-Based Cooperative Coalition Formation. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), Beijing, China, September 20-24, pp. 162–168 (2004)

[2] Amazon Mechanical Turk website (accessed, October 2007), http://www.mturk.com/mturk/help?helpPage=whatis

[3] Brooks, C.H., Durfee, E.H., Armstrong, A.: An Introduction to Congregating in Multiagent Systems. In: Proceedings of the Fourth International Conference on Multi Agent Systems (ICMAS 2000), Boston, MA, July 7-12, pp. 79–86 (2000)

[4] Faratin, P., Klein, M.: Automated Contract Negotiation and Execution as a System of Constraints. MIT, Cambridge (2001)

[5] Gaertner, D., Toni, F., Aguilar, J.: Agreeing on Institutional Goals for Multi-Agent Societies. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, Springer, Heidelberg (2008)

[6] Gaston, M.E., desJardins, M.: Agent-Organized Networks for Dynamic Team Formation. In: Proceedings of the fourth international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, Netherlands, July 25-29, pp. 230–237 (2005)

[7] Griffiths, N.: Task Delegation using Experience-Based Multi-Dimensional Trust. In: Proceedings of the fourth international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, Netherlands, July 25-29, pp. 489–496 (2005)

[8] Hanna, H., Mouaddib, A.-I.: Task Selection Problem Under Uncertainty as Decision-making. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Bologna, Italy, July 15-19, pp. 1303–1308. ACM Press, New York (2002)

[9] Jones, C., Barber, K.: Bottom-up Team Formation Strategies in a Dynamic Environment. In: AAMAS 2007 Workshop on Coordination and Control in Massively Multiagent Systems (CCMMS), Honolulu, Hawaii, USA, May 14-18 (2007)
[10] Ketchpel, S.: Forming Coalitions in the Face of Uncertain Rewards. In: Proceedings of the Twelfth National Conference on Artifical Intelligence (AAAI 1994), Seattle, Washington, July 31–August 4, pp. 414–419 (1994)
[11] Klusch, M., Gerber, A.: Dynamic Coalition Formation among Rational Agents. IEEE Intelligent Systems 17(3), 42–47 (2002)
[12] Kraus, S., Shehory, O., Taase, G.: Coalition Formation with Uncertain Heterogeneous Information. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003), Melbourne, Australia, July 14-18, pp. 1–8 (2003)
[13] Lerman, K., Shehory, O.: Coalition Formation for Large-Scale Electronic Markets. In: Proceedings of the Fourth International Conference on Multi Agent Systems (ICMAS 2000), Boston, MA, July 7-12, pp. 167–174 (2000)
[14] Martin, C.: Adaptive Decision Making Frameworks for Multi-Agent Systems. Ph.D. Thesis, University of Texas, Austin, TX (2001)
[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] Oren, N., Luck, M., Miles, S., Norman, T.: An Argumentation Inspired Heuristic for Resolving Normative Conflict. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, Springer, Heidelberg (2008)
[17] Rathod, P., desJardins, M.: Stable Team Formation Among Self-Interested Agents. In: Working Notes of the AAAI-2004 Workshop on Forming and Maintaining Coalitions in Adaptive Multiagent Systems, San Jose, California, July 26 (2004)
[18] Sandholm, T.W., Lesser, V.: Coalitions among Computationally Bounded Agents. Artificial Intelligence 94(1-2), 99–137 (1997)
[19] Sen, S., Dutta, P.: Searching for Optimal Coalition Structures. In: Proceedings of the Fourth International Conference on Multi Agent Systems (ICMAS 2000), Boston, MA, July 7-12, pp. 287–292 (2000)
[20] Shehory, O., Kraus, S.: Methods for Task Allocation via Agent Coalition Formation. Artificial Intelligence 101(1-2), 165–200 (1998)
[21] Soh, L.K., Tsatsoulis, C.: Satisficing Coalition Formation among Agents. In: Proceedings of the first international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), Bologna, Italy, July 15-19, pp. 1062–1063 (2002)
[22] Tambe, M., Pynadath, D.V., Chauvat, N.: Building Dynamic Agent Organizations in Cyberspace. IEEE Internet Computing 4(2), 65–73 (2000)
[23] Wellman, M.P., Wurman, P.R.: Market-Aware Agents for a Multiagent World. Robotics and Autonomous Systems 24(3), 115–125 (1998)

Force Versus Majority: A Comparison in Convention Emergence Efficiency

Paulo Urbano[1], João Balsa[2], Luis Antunes[2], and Luis Moniz[1]

[1] LabMAg/Universidade de Lisboa
{pub,hal}@di.fc.ul.pt
[2] GUESS/LabMAg/Universidade de Lisboa
{jbalsa,xarax}@di.fc.ul.pt

Abstract. In open societies such as multi-agent systems, it is important that coordination among the several actors is achieved efficiently. One economical way of capturing that aspiration is consensus: social conventions and lexicons are good examples of coordinating systems, where uniformity promotes shared expectations of behavior and shared meanings. We are particularly interested in consensus that is achieved without any central control or ruling, through decentralized mechanisms that prove to be effective, efficient, and robust. The nature of interactions and also the nature of society configurations may promote or inhibit consensual emergence. Traditionally, preference to adopt the most seen choices (the majority option) has dominated the emergence convention research in multi-agents, being analyzed along different social topologies.

Recently, we have introduced a different type of interaction, based on force, where force is not defined a priori but evolves dynamically. We compare the Majority class of choice update against Force based interactions, along three dimensions: types of encounters, rules of interaction and network topologies. Our experiments show that interactions based on Force are significantly more efficient (fewer encounters) for group decision making.

Keywords: Convention emergence, collective decision making, consensus, coordination, strategy update rules.

1 Introduction

Shoham and Tennenholtz [14] have argued that in multi-agent systems agents have to agree on common rules to decrease the number of conflicts and promote cooperative behavior. These rules take the form of conventions that the agents share to favor coordination. One path of research has been developed around the idea of on-line convention design where an agreement can emerge from within a population of autonomous agents by using only local available information. *"The key problem is to design a strategy update function, representing an agent's decision making process, that when used by every agent in the society, will bring the society to a global agreement as efficiently as possible"* [22]. For example, language demands agreement and convention and there has been

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 48–63, 2009.

significant research on the emergence of a common language inside multi-agent systems [6, 17, 10]. The emergence of conventions can be considered an example of Multi-Agreement Problems [9].

We are going to consider only competition between potential conventions of equal intrinsic value. Essentially, the problem can be stated as follows. A group of homogeneous agents has to decide to adopt a behavioral strategy out of a given set. There are no good reasons to prefer one strategy over the other, the individual decision, and even the collective selection are arbitrary. What is important is that everyone adopts the same strategy. We can think of examples such as the direction in which traffic flows: it is quite unimportant whether it is right or left, as long as everyone uses the same lane. Other authors have studied convention emergence considering situations where there are strategies of different intrinsic values [6, 12, 13].

In dynamical systems this problem proves intractable to solve beforehand or in a centralized fashion [16], so efforts have been concentrated in developing emergent co-learning processes that allow consensus to be achieved among all the agents (actually, usually only 90% of consensus defines convergence, to allow for isolated agents to exist without any mind-changing interactions).

Agents have a chance to change their convention when they interact with other agents, in a decentralized and locally confined manner. Shoham and Tennenholtz [15,16] have studied the efficiency of convergence in these conditions, and selected the highest cumulative reward (HCR) as the most efficient individual update rule.

In the initial research on convention emergence [14, 15, 16], there were no restrictions on interactions; any agent could interact by chance with any other individual. Kittock [7] introduced interaction graphs in order to specify restrictions on interactions and experimented with the HCR update rule over different interaction graphs. Based on his experiments with regular and fully connected graphs, he conjectured that efficiency depends on the diameter of the graph. Regarding the number of interactions needed to accomplish consensus, Kittock observed a variation with the number of agents of $O(N^3)$ for regular graphs and $O(NlogN)$ for fully connected ones.

However, regular graphs are not very realistic. *"If we pay attention to real networks, we find out that most of them have a very particular topology, they are complex networks. (. . .) Complex networks are well characterized by some special properties, such as the connectivity distribution (either exponential or power-law) or the small-world property"* [4]. Delgado et al [4, 5] have experimented with HCR update rule for fully connected, regular, scale-free and small-world graphs and their results were consistent with Kittock's, confirming the relation between efficiency and graph diameter. Delgado observed also that scale-free and small-world networks were as efficient as fully connected ones, but small-world networks were slower to converge to a unique choice.

In all of the above mentioned experiments, agents only interact through a succession of pair-wise encounters, i.e, they apply their strategy update rule after meeting one agent, randomly chosen inside the group of its "neighbors". Walker

and Wooldridge [22] introduced the deterministic Simple Majority (SM) strategy update rule, adapted to simultaneous encounters, where each agent has access to the strategies of all its neighbors before strategy update. Delgado [4] introduced a stochastic variation to SM, named the Generalized Simple Majority (GSM).

In [21] a new rule for strategy update was introduced, named Recruitment based on Force with Reinforcement (RFR). This rule showed faster convergence than HCR in the case of fully connected networks. In RFR, agents have different power to influence others, but their force is not defined a priori in a hierarchical network as in [8], rather it evolves dynamically as a result of interactions. Agents submit to stronger agents, copying their strategies, but also inheriting their force, in a double mimetic process where there is also a reinforcement process whenever two agents with the same strategy meet. Force reflects the agents' perception of their strategy diffusion in the population, through their experience and the experiences of the agents they have interacted with. Experience (strategy + force) is transferred between agents by mimicry. We have applied RFR to the coordination of artificial agents for the generation of swarm paintings [19]. In another work, where we considered the co-existence of several concomitant social networks, and mechanisms permitting permeability (some kind of information transfer) among the contexts in different networks, convergence is not always guaranteed. But RFR has allowed for a more frequent and quicker convergence that were not possible in some of the hardest cases (networks types in which convergence occurs fewer times) [2, 3].

In [20] we compared Recruitment based on Force against HCR for different topology graphs and RFR proved to be more efficient for Fully connected, Small-world, Scale-free, Regular and Random networks.

In the present paper, we advance research on the emergent collective adoption of a common strategy. We expand and summarize our results concerning RFR versus HCR/EM, and introduce a new behavior called Recruitment of the Strongest with Reinforcement, adapted to simultaneous encounters.

This paper is organized as follows: in the next section we describe the different network topologies used in the experiments. Then we introduce two strategy update rules (one based on force, and the other on majority) for pair-wise encounters and compare them across the different social graphs. Then we continue by describing strategy update rules (one based on force, and the other on majority) for simultaneous interactions, which are compared. Finally we conclude pointing some future directions. Since this works extends previous work presented in [20] it is unavoidable to describe the results of that research here.

All experiments were conducted using the most recent version of the Netlogo platform [25], version 4.0.2, released in December 2007.

2 Interaction Graph Topologies

An interaction graph is a general way of modeling restrictions on interactions. Restrictions could be due to spatial barriers, communicating links, different castes, social groups, etc. We have experimented with five network topologies:

fully connected, regular, scale-free, small-world and random. These topologies cover a wide spectrum of values regarding three important network properties: average path length, network diameter and clustering coefficient.

The average path length is calculated by finding the shortest path between all pairs of nodes, adding them up, and then dividing by the total number of pairs. This indicates, on average, the number of steps it takes to get from one member of the network to another. The diameter of a graph is the longest shortest-path between nodes. The clustering coefficient is a measure of "all-my-friends-know-each-other" property. When it is high, we may say: "the friends of my friends are my friends." The clustering coefficient of a node is the ratio of existing links connecting a node's neighbors to each other to the maximum possible number of such links. The clustering coefficient for the entire network is the average of the clustering coefficients of all the nodes.

2.1 Regular Graphs

By definition, a graph is considered regular when every node has the same number of neighbors. We are going to use a special kind of regular graph, explored in [7] and named Contract Net with Communication Radius K in [18]. $C_{N,K}$ is the graph (regular ring lattice) on N nodes such that node i is adjacent to nodes $(i+j)\ mod\ N$ and $(i-j)\ mod\ N$ for $1 \leq j \leq K$. In a $C_{N,K}$ graph, every node has connectivity $2 * K$. These are highly clustered graphs but have very long path lengths (average path length and diameter grow linearly with the number of nodes).

2.2 Fully Connected Graphs

In this type of graph topology, named K_N, there are no restrictions on the pattern of interactions: each agent is connected to every other agent in the society. This means that an agent can potentially interact with any other agent. K_N is a special case of a regular graph where each agent has $N-1$ neighbors, in a group of N agents.

2.3 Random Graphs

$R_{N,K}$ are random graphs with N nodes and average connectivity of K. Every node, has on average, K neighbors chosen randomly. The clustering coefficient of $R_{N,K}$ tends to 0 and the average path length is small and grows logarithmically with N.

2.4 Scale-Free Graphs

This network type, $S_{N,\gamma}$ has a large number of nodes connected only to a few nodes and a small number of well-connected nodes called hubs. The power law distribution highly influences the network topology. It turns out that major hubs are closely followed by smaller ones. These ones, in turn, are followed by other

nodes with an even smaller degree, and so on. As the network changes in size, the ratio of hubs to the number of nodes in the rest of network remains constant — this is why it is named scale-free. The connectivity of a scale-free network follows a power law $P(k) \sim k^{-\gamma}$. Such networks can be found in a surprisingly large range of real world situations, ranging from the connections between websites to the collaborations between actors.

To generate the scale-free graphs we have used the Albert and Barabsi extended model [1], since Delgado argues that it allows some control over the exponent (γ) of the graph [4]. The inspiration of this algorithm is that of "preferential attachment", meaning that the most "popular" nodes get most of the links. The construction algorithm relies on four parameters: m_0 (initial number of nodes), m (number of links added and/or rewired at every step), p (probability of adding links), and q (probability of edge rewiring). The algorithm starts with m_0 isolated nodes and at each step performs one of these three actions until the desired number N of nodes is obtained:

1. with probability p, add m ($\leq m_0$) new links. Pick two nodes randomly. The starting point of the link is chosen uniformly and the end point of the link is chosen according to the probability distribution:

$$\Pi_i = \frac{(k_i + 1)}{\sum_j (k_j + 1)}$$

   where Π_i is the probability of selecting the ith node and k_i is the number of edges of node i. This process is repeated m times.
2. with probability q, m edges are rewired. That is, repeat m times: choose uniformly at random one node i and one link l_{ij}. Delete this link and choose a different node k with probability $\{\Pi_l\}_{l=1,\ldots,N}$ and add the new link l_{ik}.
3. with probability $1-p-q$ add a new node with m links. These new links will connect the new node to m other nodes chosen according to $\{\Pi_l\}_{l=1,\ldots,N}$. Using this algorithm, the parameter γ is a function of m and p:

$$\gamma = \frac{(2m(1-p)+1)}{m+1}$$

2.5 Small-World Graphs

The Small World graphs are highly clustered graphs (like regular graphs) with small average path lengths (like random graphs, described above). To generate small world graphs we use the Watts-Strogatz model [24, 23]. It depends on two parameters, connectivity (K) and randomness (P), given the size of the graph (N).

This model starts with a $C_{N,K}$ graph and then every link is rewired at random with probability P, that is, for every link l_{ij} we decide whether we change the "destination" node with probability P; if this is the case, we choose a new node k uniformly at random (no self-links allowed) and add the link l_{ik} while erasing link l_{ij}. In fact, for $P = 0$ we have $W_N = C_{N,K}$ and for $P = 1$ we have a completely

random graph (but not scale-free). For intermediate values of P there is the "small-world" region, where the graph is highly clustered (which means it is not random) but with a small characteristic path length (a property shared with random graphs).

Albert-Barabsi model graphs do not have the small-world property and reciprocally the Watts-Strogatz model does not generate scale-free graphs (it generates an exponential connectivity distribution, not a power law).

3 Strategy Update Rules in Pair-Wise Interactions

Agent societies consist of N agents on a graph, where every agent is located on a node of the graph. Its adjacent nodes are its neighbors. In order to make experiments and simulations we have adopted a simple agent model where they have at their disposal a finite repertoire of strategies. We only deal with the two strategies case. We use here the concept strategy in a very abstract way: it can be a social norm, like driving on the left or on the right lane, the meaning of a word, an orientation for flocking, etc. In order to focus on the essential features of agent interactions, the agent environment consists solely of other agents, which in turn depend on the network topology. So, each agent has to adopt one of the strategies from the repertoire and through mutual interactions they can change their adopted strategies over time. A consensus, or collective choice, exists when all the agents are using the same particular strategy.

From the point of view of each agent, there is an interaction scenario of a sequence of pair-wise asymmetric encounters, where it meets randomly one of its neighbors. After an encounter, each agent updates its strategy, i.e., it selects the strategy it will use in the next interaction — the result need not necessarily be a change in strategy adoption. Therefore, agents need strategy update rules (behaviors). We assume that each agent updates its strategy at each encounter. Shoham and Tennenholtz [14, 16] have studied the effects of updating less frequently on the efficiency of global choice emergence. We only consider asymmetric encounters, where only one of the agents applies its strategy update rule, based on the strategies used by all the individuals involved in the interaction. Thus, interactions are always considered from the point of view of one particular agent. We now describe the two strategy update rules whose performance we subsequently compare. In the first scenario the agent and its selected partner strategies are crucial for the update, and in the second case, it is the simultaneous strategies of its neighbors and its own strategy that influence the update rule.

3.1 External Majority/Highest Cumulative Reward/Feedback Positive with Score

The External Majority strategy update rule (EM) was introduced by Shoham and Tennenholtz [14] and is the following: adopt the strategy that was observed more often in other agents in the last m interactions, and remain with your

current strategy otherwise — in case of a draw do not change. In EM, a memory of size m is used to register the strategies observed during the last m interactions. An agent updates its memory after observing its partner strategy and then decides to change to a new strategy only in case it was more frequently observed than the current one.

In the context of lexical emergence, Kaplan [6] introduced a strategy update rule called Positive Feedback with Score, which is pretty much the same as EM. The only difference is that, in case of equality, the agent does not necessarily remain with its current strategy but chooses randomly one of the previously most seen strategies. Kaplan considered the full history of encounters for strategy update.

Probably the most cited strategy update rule is the Highest Cumulative Reward update rule (HCR), which was developed in the context of game theory by Shoham and Tennenholtz [15]. Intuitively, a game involves a number of players each of which has available to it a number of strategies. Depending on the strategies selected by each agent, they each receive a certain payoff. The payoffs are captured in a payoff matrix.

Thus, returning to the context of this paper, when two agents meet they play a pure coordination game, which is an instance of the class of coordination games introduced by Lewis [11]. The pure coordination game is defined by the following symmetric payoff matrix:

	A	B
A	+1,+1	-1,-1
B	-1,-1	+1,+1

Suppose that every player has two available strategies, say A and B. If both players play A, both players receive a payoff of 1. If they play B they receive a payoff of 1. When the players do not agree, for example, player 1 plays A and player 2 plays B, they will both receive a payoff of -1; the remaining situation is symmetric. The condition on the entries of the payoff matrix makes it clear that the best action consists in playing the same strategy, i.e., coordinating.

According to the HCR update rule, an agent switches to a new strategy if and only if the total payoff obtained from that strategy in the latest m interactions is greater than the payoff obtained from the current strategy in the same last m interactions. The m parameter may not have limit, implying that the full history of pair-wise meetings will play a role in the strategy selection process, or we can implement a forgetting mechanism by limiting m. The agents' memories register the payoffs that each strategy has received during the last m encounters. When an agent receives new feedback it discards its old memory to maintain the memory at a fixed size.

Shoham and Tenneholtz [16] showed that EM and HCR are equivalent strategy update rules in the case there is a repertoire of two strategies from which to select.

3.2 Recruitment Based on Force with Reinforcement

In this strategy update rule there is a new attribute, besides the strategy, called force. Thus, agents are characterized by two attributes: strategy and force. Both attributes can be observed during encounters and are subject to imitation. There are three main mechanisms incorporated in RFR behavior. During interactions (1) weaker agents always adopt the strategies of the stronger ones; (2) weaker agents always adopt the force of the stronger ones, independently of the strategies involved; and (3) there is a positive reinforcement when an agent faces another with the same strategy, which is the amplification mechanism for strategy diffusion.

During a dialogue (asymmetric), involving two agents, one is the observing agent and the other is the observed one. The observing agent "fights" metaphorically with its partner, comparing its force with the partner's force. If the observing agent is the stronger one, or if they have identical force, it will loose the fight; otherwise it will be the winner. The winner's behavior is: (1) if they have the same current strategy its force is reinforced by 1 unit, otherwise (2) it does nothing. The loser behavior is: (1) it imitates both strategy and force in case they have different current strategies, otherwise (2) imitates the force of the winner agent and increments its force by 1 unit. In summary, stronger agents recruit weaker agents for their parties, enlarging the influence of their options. As the recruited agents will be at least as strong as the winners, they will be better recruiters. The more the strategy is diffused the more it will have stronger representatives. This is due to the behavior's reinforcement and imitation components. So when a player observes a stronger agent it is recruited, by adopting the stronger strategy, inheriting its force, i.e., updating the information about the strategy it is now adopting.

The force attribute can be interpreted not as the strength of an individual because, being imitated, it is diffused along agents, and it is not private to any agent, but as the force of the strategy the player is adopting, seen from its local history of interactions. The force of an individual reflects the diffusion strength of the strategy adopted and this diffusion strength is transferred and amplified between individuals. RFR possesses something extra when compared with EM. In EM, agents just register their partners strategies while in RFR they can transfer their experiences, exchanging the diffusion strength of their strategies. The idea of exchanging of experiences was introduced in [14], in the context of HCR where agents exchange their memories.

At the beginning of an experiment, we have to face a situation of maximal competition: every agent has the same value of force (0) and each agent adopts randomly one of two strategies. Their forces and strategies will evolve along with interactions. Therefore, there is no a priori (off line) power hierarchy. This shows a clear contrast with the work of Kittock on authority [8], where agents have fixed different influences on one another's behavior, modeled by the probability of receiving feedback during encounters: the more influential agents receive feedback with some probability, while the less influential agents always receive feedback. RFR does not try to simulate any natural behavior and it was introduced in [21]

as the best outcome of experimenting with several strategies involving the idea of emergence of hierarchies in consensus emergence. We think it is simpler than the EM as agents do not need to maintain the recent history of encounters in spite of using force as an extra attribute. It maintains the essential properties of EM, which is the capability to adapt, locality (an agent relies only is the information gathered in interactions), and no more cognitive skills than the capability to imitate.

4 Experiments and Results for the Pair-Wise Strategy Update Rules

The system starts with half of the agents adopting randomly one of the strategies (50% possibilities for each). At each step, every agent, in an asynchronous way, is selected and chosen for asymmetric strategy updating. The order of selected agents is completely random and changes in each iteration. (In fact, it is the natural scheduling of Netlogo agents, but we think that the order of agents for strategy update is not an issue, because strategies are randomly adopted initially).

We use in this paper the same measure of performance as in [4, 7]: average number of interactions to a fixed convergence, where convergence means the fraction of agents using the majority strategy. We made 100 runs for each parameter setting and in each run we have measured the number of encounters until 90% convergence and calculated the average performance of the different runs.

Our main goal was to compare the performance of the two strategy update rules: External Majority (EM) and Recruitment based on Force with Reinforcement (RFR). In order to choose the size of memory (parameter m, see 3.1) of the EM update rule we have conducted many experiments with different types of networks. The best performances were obtained with $m = 3$. Both Kittock [7,8] and Delgado [4,5] used a memory size of 2 in their convention experiments, but size 3 EMs out-performed size 2 EMs in our own tests. So we will only present the comparison between RFR and the best EM (EM with a memory size of 3). The comparison between these two behaviors was made over the different kinds of networks described earlier, using different parameter settings. Again, we only present here the most representative experiments.

For all settings we made the number of agents range from 100 to 1000, using a step of 100.

In figure 1, we can see a comparison between the average number of meetings needed for a 90% convergence using the two behaviors in fully connected (dashed lines) and random networks ($K = 20$, solid lines), and with RFR (diamonds) and EM (squares) as update rules. In both cases RFR clearly outperforms EM, since consensuses are reached in a much lower number of meetings. Besides, the difference appears to increase with N. In fact, in the above experiments, improvements vary between 26% and 37%.

Using other types of networks, the difference between the two behaviors is even clearer. In figure 2 we compare the performance of the two behaviors in

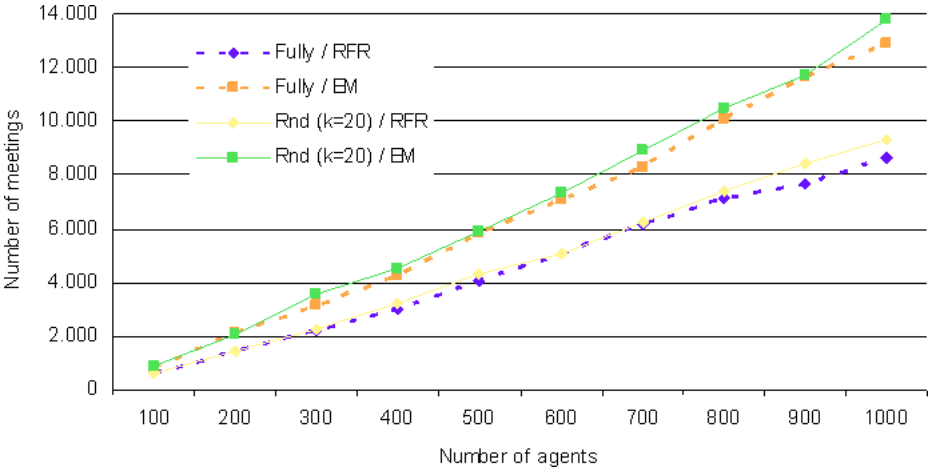


Fig. 1. RFR vs EM in random networks (with 40 neighbors per agent on average — $R_{N,20}$) and fully connected networks (K_N)

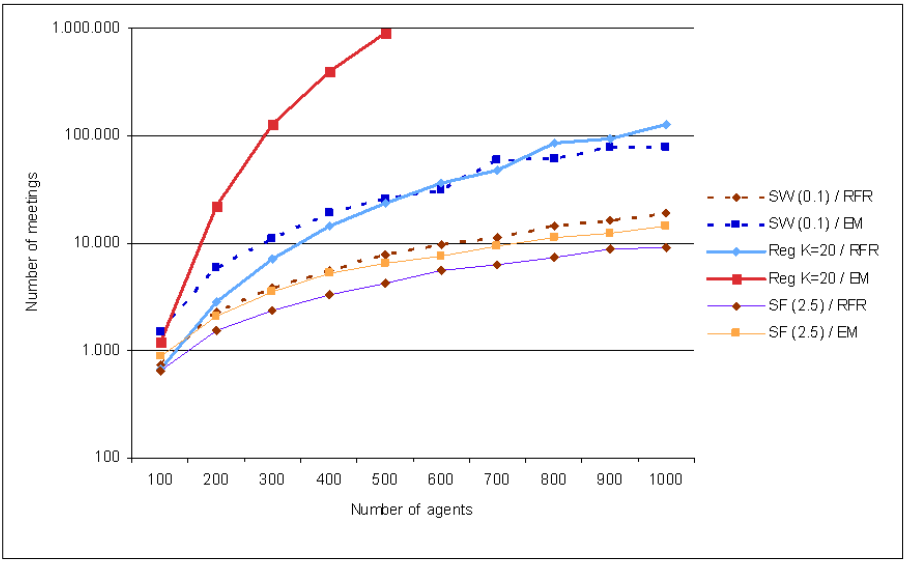


Fig. 2. RFR vs EM in small-world, $W_N(K = 12; P = 0, 1)$, regular, $C_{N,20}$, and scale-free networks, $S_{N,2.15}$

small-world networks (with $P = 0.1$, dashed lines), regular networks (with $K = 20$, each agent with 40 neighbors, solid lines) and scale-free networks (with $\gamma = 2.5$, dotted lines). In these cases we need to use a logarithmic scale in the y axis. The average number of meetings needed to reach a consensus is again much lower when RFR (diamonds) is used. In Scale-free networks, the performance is

improved between 25% and 37%; in Small-world networks improvements reach 80%. Using regular networks the difference with N greater than 500 is so huge that it is not represented.

5 Strategy Update Rules in Simultaneous Interactions

Now, in a clear contrast with previous experiments, an agent interacts, in each encounter, not only with a randomly chosen neighbor but with all its neighbors at the same time. Interaction is again asymmetric, seen from the point of view of an agent — the simultaneous state of neighbors is used to update agents' strategy, and naturally will have to be taken into account in the strategy update rules. Again, each agent updates its strategy at each encounter.

5.1 Generalized Simple Majority

The natural strategy update rule to use in simultaneous interactions with all neighbors, equivalent to EM, for the pair-wise encounter, would be the Simple Majority created by Walker and Wooldridge [22]. Following this rule, an agent only changes strategy when more than half of the neighboring agents adopt a different strategy than its current one. This deterministic rule does not guarantee convergence to a consensual situation for some networks, especially the regular ones. In some cases, even in small groups, agents get stuck in a deadlock, never reaching a situation of full convergence or even 90% convergence. Delgado [4] developed a stochastic version of Simple Majority, which they named the Generalized Simple Majority (GSM). We are going to describe GSM, but not exactly the same way as was originally defined by Delgado.

Suppose we have N agents in an interaction graph. If agent j has K neighbors it will adopt state S with a probability that depends on the number of neighbors adopting S (K_S):

$$f_\beta(K_S) = \frac{1}{1 + e^{2\beta(2(1-K_S/K)-1)}}$$

The formula above is a corrected version of the one introduced by [4]. This rule generalizes Simple Majority since, for $\beta \to \infty$, an agent adopts state S only when at least half of the neighbors are in that state. Note that when a tie occurs, the probability of changing is 50%. In our experiments, we used $\beta = 10$, which is was also used in [4].

As illustrated in figure 3, the probability of adopting S is positive even when the neighbors adopting S are not in majority. This is true for values greater than 38%. Conversely, it is possible to adopt a state that is adopted by the minority of the neighbors, which can happen if the majority is no greater than a percentage of 62%. Note that when a tie occurs, the probability of changing is 50%. Delgado provided some analytical evidence for the convergence of GSM, but no theorem exists that guarantees it. In our experiments, GSM has always converged to a consensual situation of 90%, in all types of networks, varying the number of agents until 1000.

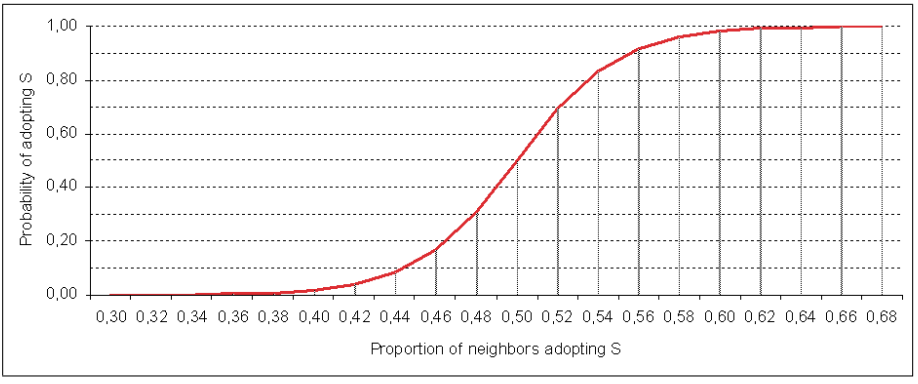


Fig. 3. Probability of changing to a new state S, for $\beta = 10$, given the percentage of neighbors that are adopting S

5.2 Recruitment by the Strongest with Reinforcement

In order to adapt the RFR update rule to simultaneous interactions, partners in encounters will not be chosen randomly as before. Now an agent can access the strategies of all its neighbors — it will choose the strongest to interact with in a pair-wise way. The interaction behavior is pure Recruitment based on Force with Reinforcement (RFR), but with the strongest of its neighbors, not with a randomly chosen neighbor. We name this strategy update rule Recruitment by the Strongest with Reinforcement (RSR). There is simultaneous access to all neighbors but actual interaction (applying RFR) is only with the strongest. In the RSR behavior agents begin with the same force value zero.

6 Experiments and Results for the Simultaneous Strategy Update Rules

The comparison between RSR and GSM was made over the different kinds of networks described in section 2, using different parameter settings. Again, we only present here the most representative experiments and for all settings we made the number of agents range from 100 to 1000, using a step of 100. We counted the number of encounters necessary for 90% consensus along 100 runs for each parameter setting and values were averaged.

In figure 4, we can see a comparison between the average number of meetings needed for a 90% convergence using the two behaviors in random networks ($K = 20$, dotted lines) and scale-free networks (with $\gamma = 2.15$, solid lines) with RSR (squares) and GSM (diamonds) as update rules.

RSR clearly outperforms GSM, since consensuses are reached in a much lower number of meetings (the improvement in regular networks is remarkable).

In the fully connected case, GSM is slightly better, around 1%, which is marginal and would not be perceptible in the chart. The results are

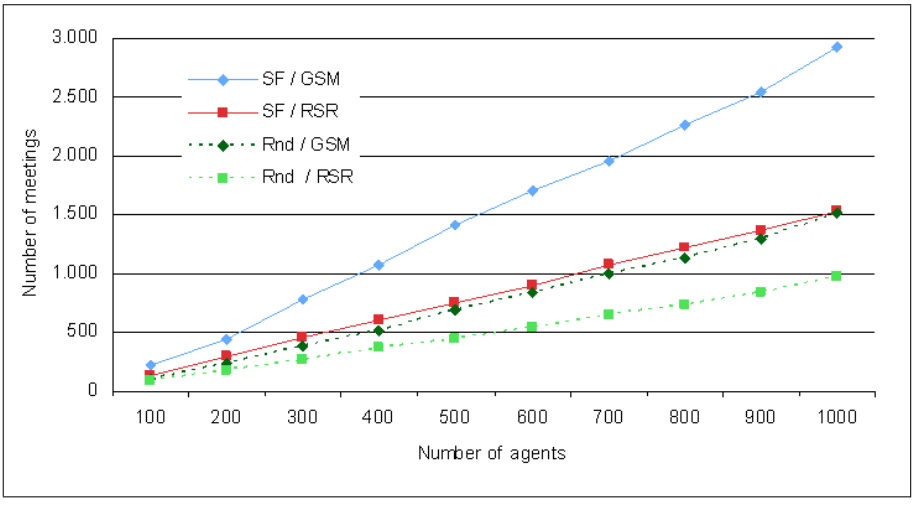


Fig. 4. RSR vs GSM in random networks (with 40 neighbors per agent on average — $R_{N,20}$) and scale-free networks, $S_{N,2.15}$.

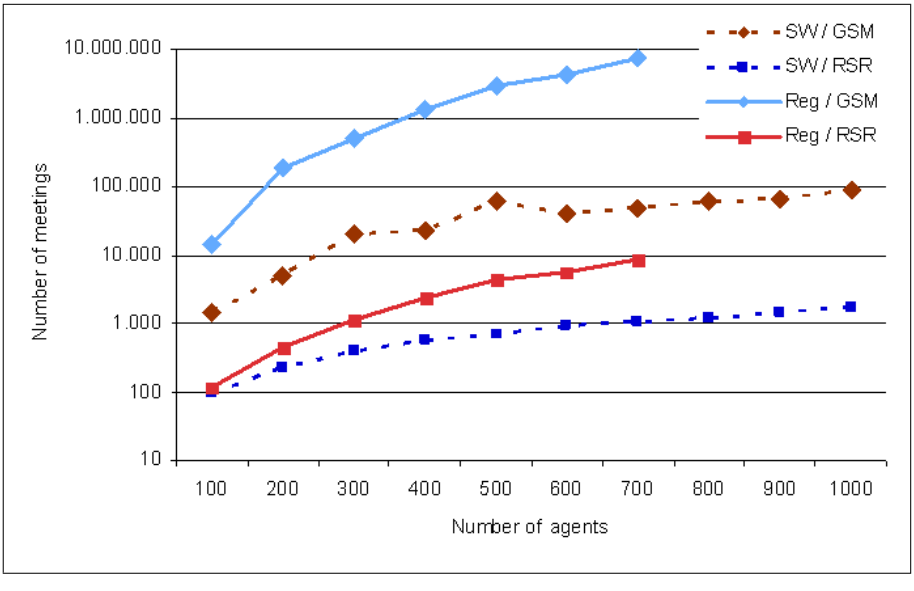


Fig. 5. RSR vs GSM in small-world, $W_N(K = 12; P = 0.1)$, and regular networks, $C_{N,20}$

understandable in this type of networks, because as initially every agent chooses its strategy in a random way, there will be almost certainly a majority and every agent will chose the winning strategy in just one encounter. On average, we

won't need N interactions, because unless the last agent belongs to the minority, consensus will be attained earlier (in less encounters). This way, it would be very hard to beat GSM in fully connected networks, but never the less, RSR attained a very good result (only 1% later). Note that we are using time to refer to the number of interactions needed to achieve consensus.

In the Random networks case, we have an improvement for RSR around 16% for $N = 100$ and gradually the difference increases, and stabilizes around 35% for N greater than 500.

In the Scale-free case, RSR, again, exceeds GSM, with an improvement that varies between 33% and 48% ($N > 500$).

Using other types of networks, the difference between the two behaviors is even clearer. In figure 5, we compare the performance of the two behaviors in small-world networks (with $K = 12$, $P = 0.1$, dashed lines) and regular networks (with $K = 20$, solid lines). In these cases we need to use a logarithmic scale in the y axis. The average number of meetings needed to reach a consensus is now much lower when RSR (squares) is used. In Regular networks, the performance is improved in 99%, reaching 99,9% for N above 600; in Small-world networks improvements reach 93% for $N = 100$, reaching 98% for N bigger than 300.

7 Conclusions and Future Work

Regarding our main goal of comparing strategy update rules based on majority against ones based on force, the main conclusion is that those based on force almost always perform better (converge in fewer interactions) than the ones based on majority. This conclusion is valid for all types of networks with different parameter settings, except the case of fully connected graphs in simultaneous encounters. Even considering that we did not exhaust all type of networks and its parameters, this is an impressive result. According to our experiments, strategy update rules based on force, show at least a 25% improvement over the majority one, but is much higher in many settings, and extreme in regular networks. An interesting point is that our results for the RFR strategy update rule show that the network diameter strongly influences the performance, as noted in other settings by authors such as Kittock [7] and Delgado [4]. The smaller the network diameter is, the better. Also, the performance of fully connected networks and scale-free ones seems to be quite similar (as can be observed in figure 1, comparing lines with similar marks, that correspond to the same behavior). This is a very important result since scale-free networks are much less expensive than fully connected ones. Nevertheless, we must perform experiments with greater values of N in order to obtain definite conclusions on this matter. Besides performing experiments with larger values of N, three other aspects are scheduled for short-term future work. One is to let the agents choose between more than two strategies. The other is to explore the performance of these behaviors in networks with a dynamical structure, and finally we want to see force strategy update rules applied to situations where agents can choose between differently-valued strategies.

Acknowledgements. We would like to thank the reviewers for their careful revision of the paper and for their useful comments.

References

1. Albert, R., Barabási, A.: Topology of evolving networks: local events and universality. Physical Review Letters 85, 5234–5237 (2000)
2. Antunes, L., Balsa, J., Urbano, P., Coelho, H.: Exploring context permeability in multiple social networks. In: Proceedings of the World Congress on Social Simulation (2008)
3. Antunes, L., Balsa, J., Urbano, P., Coelho, H.: The challenge of context permeability in social simulation. In: Proceedings of the Fourth European Social Simulation Association (2007)
4. Delgado, J.: Emergence of social conventions in complex networks. Artificial Intelligence 141(1/2), 171–185 (2002)
5. Delgado, J., Pujol, J.M., Sangüesa, R.: Emergence of coordination in scale-free networks. Web Intelligence and Agent Systems 1(2), 131–138 (2003)
6. Kaplan, F.: L'Emergence D'un Lexique Dans Une Population d'Agents Autonomes. PhD thesis, Université de Paris 6 (2000)
7. Kittock, J.E.: Emergent conventions and the structure of multi-agent systems. In: Nadel, L., Stein, D. (eds.) Studies in the Sciences of Complexity. 1993 Lectures in Complex Systems. Addison-Wesley, Reading (1995)
8. Kittock, J.E.: The impact of locality and authority on emergent conventions: Initial observations. In: Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, WA, pp. 420–425 (1994)
9. Lakkaraju, K., Gasser, L.: A unified framework for multi-agent agreement. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, p. 21, IFAAMAS (2007)
10. Lakkaraju, K., Gasser, L.: Norm Emergence in Complex Ambiguous Situations. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428. Springer, Heidelberg (2008)
11. Lewis, D.K.: Convention: A Philosophical Study. Harvard University Press, Cambridge (1969)
12. Pujol, J.M., Delgado, J., Sangüesa, R., Flache, A.: The role of clustering on the emergence of efficient social conventions. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI 2005, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, pp. 965–970. Professional Book Center (2005)
13. Salazar, N., Rodriguez-Aguilar, J.A., Arcos, J.L.: Infection-based self-configuration in agent societies. In: Ebner, M., Cattolico, M., van Hemert, J., Gustafson, S., Merkle, L.D., Moore, F.W., Congdon, C.B., Clack, C.D., Moore, F.W., Rand, W., Ficici, S.G., Riolo, R., Bacardit, J., Bernado-Mansilla, E., Butz, M.V., Smith, S.L., Cagnoni, S., Hauschild, M., Pelikan, M., Sastry, K. (eds.) GECCO 2008 Workshop: Evolutionary Computation and Multi-Agent Systems and Simulation (ECoMASS), Atlanta, GA, USA, July 12-16, pp. 1945–1952. ACM, New York (2008)
14. Shoham, Y., Tennenholtz, M.: Emergent conventions in multi-agent systems. In: Rich, C., Swartout, W., Nebel, B. (eds.) Proceedings of Knowledge Representation and Reasoning (KR&R 1992), pp. 225–231 (1992)

15. Shoham, Y., Tennenholtz, M.: Co-learning and the evolution of social acitivity. Technical Report CS-TR-94-1511, Stanford University, Department of Computer Science (1994)
16. Shoham, Y., Tennenholtz, M.: On the emergence of social conventions: Modeling, analysis and simulations. Artificial Intelligence 94(1-2), 139–166 (1997)
17. Steels, L.: The origins of ontologies and communication conventions in multi-agent systems. Autonomous Agents and Multi-Agent Systems 1(2), 169–194 (1998)
18. Tennenholtz, M.: Convention evolution in organizations and markets. Computational & Mathematical Organization Theory 2(4), 261–283 (1996)
19. Urbano, P.: Consensual paintings. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 622–632. Springer, Heidelberg (2006)
20. Urbano, P., Balsa, J., Antunes, L., Moniz, L.: Efficiency of the emergence of consensus in complex networks - assessing force influence. In: Costa, A.R. (ed.) Proceedings of the BWSS/SBIA Workshop (2008)
21. Urbano, P., Coelho, H.: From consensus to consensus: Random evolution of collective choices. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005, pp. 274–279. IEEE, Los Alamitos (2005)
22. Walker, A., Wooldridge, M.: Understanding the emergence of conventions in multiagent systems. In: Lesser, V.R., Gasser, L. (eds.) Proceedings of the First International Conference on Multiagent Systems (ICMAS), June 12-14, pp. 384–389. MIT Press, San Francisco (1995)
23. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998)
24. Watts, D.J.: Small Worlds: The Dynamics of Networks Between Order and Randomness. Princeton University Press, Princeton (1999)
25. Wilensky, U.: NetLogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL (1999)
`http://ccl.northwestern.edu/netlogo/`

Automatic Generation of Distributed Team Formation Algorithms from Organizational Models

Michael Köhler-Bußmeier and Matthias Wester-Ebbinghaus

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D–22527 Hamburg
{koehler,wester}@informatik.uni-hamburg.de

Abstract. Software systems are subject to ever increasing complexity and in need of efficient structuring. The concept of organization as an expressive and abstract real-world reference presents a promising starting point. Organizational concepts have particularly been studied within the multi-agent systems community. However, there exists a conceptual gap between organizational specifications and their multi-agent implementation. We address this problem by presenting an integrated approach to formalize organizational models with Petri nets and to directly deploy these specifications in a multi-agent system. The operational semantics of Petri nets establishes a close link between organizational specification and deployment that eases system development and maintenance. As an important example, we are able to describe the formation of multi-agent teams in an organizational scenario in terms of Petri net dynamics.

1 Introduction

Inspiration from organization theory serves to handle phenomena in multi-agent systems that carry a *supra-individual* character. Especially in the past few years multi-agent system researchers have begun to comprehend the organizational metaphor as the central instrument to combine local agent autonomy with reliability and predictability on the system level (we provide a detailed overview of recent and current work concerning modelling approaches, development methodologies and middleware in [1]). This combination is achieved by imposing "organizational facts" onto the system. Boissier [2] identifies several organizational dimensions that different approaches incorporate to varying degrees. Among these dimensions, the most prominent are the structural, functional and interactional (dialogical) dimensions. They respectively carry specifications concerning the structure of the collective level of a multi-agent system, its global functioning and the interactions between agents in terms of communication.

One central questions is how to close the conceptual gap between high-level organizational specifications of a software system to be and its multi-agent system implementation. The above mentioned organizational dimensions are not mutually independent. Only in concert they draw a coherent picture. Thus agents

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 64–79, 2009.

are embedded in a system of interrelated organizational specifications and left with the task of operationalizing them. One possibility to deal with this problem is to introduce organizational dimensions with the specific purpose to establish an explicit link between other dimensions. For example, $\mathcal{M}$OISE$^+$ [3] and ISLANDER [4] exhibit *deontic* and *normative* dimensions respectively. These enrich an agent's structural embedding with obligations and permissions towards functional and dialogical specifications. Some approaches with a particular emphasis on *teamwork* go even further. They link agents to an organizational framework in terms of the agents' (joint) *mental states* [5,6]. Depending on the structural embedding of the agents (which roles they occupy) team and team plan formation are carried out by establishing mutual beliefs, joint commitments and joint intentions concerning the following joint team activity.

In this paper we present a related but distinctive approach. We propose an approach to model organizations based on Petri nets. It integrates a structural, a functional and an interactional dimension. However, the peculiarity of using Petri nets as a modelling technique renders additional dimensions or mechanisms for paving the way to operationalization obsolete. The operational semantics of Petri nets allows them to be directly utilized in a software implementation. We exemplify this idea by defining teams solely in terms of Petri net dynamics and specifically elaborate on the case of team formation in an organizational setting. In Section 2 we present our formal Petri net model of organizations. In Section 3 we demonstrate how the formal specifications may be instantiated in the form of a multi-agent system and how they directly lead to a distributed algorithm for team formation to be carried out by the agents. In Section 4 we give full particulars on the role of each individual agent in the course of the team formation procedure and thus link the organizational coordination plans with individual action plans. We conclude our results in Section 5 and relate them to the wider context of our current work.

2 Formal Model of Organizations

We present a Petri net model of organizations. It is intentionally lean and open to be extended or specialized for specific purposes. Our model focuses on the two fundamental (and opposing) requirements for every organized activity postulated by Mintzberg [7], the *division of labour* into various tasks to be performed and the *coordination* of carrying out these tasks in order to accomplish the overall activity. Accordingly, our model basically includes *tasks*, *roles* included in the accomplishment of these tasks and (hierarchical as well as horizontal) *task-based relationships between roles* for the coordination. Our Petri net-based approach to formalize these concepts allows to elegantly combine both the structural (through the structure of a Petri net model itself) and the behavioural (through the operational semantics of Petri nets) properties of organizational models.

In this section we address the *formal organization* without reference to the organization's members [8]. It is not until Section 3 that we turn to multi-agent system deployment and thus to the *informal organization* with reference to agents as members of the organization.

Petri nets offer both a graphical representation and formal semantics. In [9] we present our model in more detail together with formal theorems and proofs. In this paper we just assume a general understanding of Petri nets in terms of places, transitions and arcs as their basic elements. For a thorough introduction into Petri nets we refer to [10].

Definition 1. *A* Petri net *is a tuple $N = (P, T, F)$ where P is a set of places, T is a set of transitions, disjoint from P, i.e. $P \cap T = \emptyset$, and $F \subseteq (P \times T \cup T \times P)$ is the flow relation.*

The preset *of a node y is ${}^\bullet y := (\_\ F\ y)$ and* postset *is $y^\bullet := (y\ F\ \_)$. For a set $A \subseteq P \cup T$ we define ${}^\bullet A = \bigcup_{a \in A} {}^\bullet a$ and $A^\bullet = \bigcup_{a \in A} a^\bullet$.*

The minimal nodes are ${}^\circ N = \{x \in P \cup T | {}^\bullet x = 0\}$, the maximal are $N^\circ = \{x \in P \cup T | x^\bullet = 0\}$.

A finitely branching Petri net $N = (P, T, F)$ is called a causal net *iff the transitive closure F^+ is acyclic and for all $p \in P$ the conditions $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$ hold.*

Turning to Petri net dynamics, we have to consider markings and transition firing. We define the mappings $\mathbf{pre}, \mathbf{post} : T \to (P \to \mathbb{N})$ by $\mathbf{pre}(t)(p) := |F \cap \{(p,t)\}|$ and $\mathbf{post}(t)(p) := |F \cap \{(t,p)\}|$. A marking of a Petri net (P, T, F) is a multiset (or: a vector) of places: $m \in \mathbb{N}^P$. A transition $t \in T$ of N is *enabled* in marking m iff $m(p) \geq \mathbf{pre}(t)(p)$ holds for all $p \in P$. The successor marking m' obtained after firing t is defined $m'(p) = m(p) - \mathbf{pre}(t)(p) + \mathbf{post}(t)(p)$. Firing is denoted $m \xrightarrow{t} m'$. The notation extends to firing sequences the usual way. The set of all *reachable markings* is $RS(m) = \{m' \mid m \xrightarrow{*} m'\}$.

2.1 Roles and Services

We begin by introducing roles and services in the context of our model.

Roles describe specific expectation structures with respect to the behaviour of agents. Technically speaking, roles are some kind of type for an agent describing its behaviour. Given a set of *atomic roles Rol* the set $\mathcal{R} := 2^{Rol} \setminus \emptyset$ is the role universe. Each $R \in \mathcal{R}$ is called a *role*. The structure of roles is modelled by the partially ordered set $(\mathcal{R}, \subseteq)$. The singletons sets $\{r\}$ are identified with r itself.

Services in our model are accompanied by *service nets* that represent the model's interactional dimension. A service net is a Petri net that describes the interactions that take place between agents in order to carry out the corresponding service. Each transition models a task and places connect tasks in order to form life lines and message exchanges. However, service nets abstract away from particular agents and identify the interacting roles instead. We assume a unique assignment of roles to service nets.

Definition 2. *A* service net *$D = (N, r)$ is a Petri net $N = (P, T, F)$ with a role assignment $r : T \to Rol$. Define $R(D) := r(T)$.*

Given a service net D and a role $R \subseteq R(D)$ we can restrict D to the subnet $D[R] = (P_R, T_R, F_R)$ of D, called the R-component of D, defined by the nodes related to R: $T_R := r^{-1}(R)$ and $P_R := ({}^\bullet T_R \cup T_R {}^\bullet)$.

A set of service nets $\mathcal{D}$ *is called* service universe*, whenever* $R \in R(D_1) \cap R(D_2)$ *for* $D_1, D_2 \in \mathcal{D}$ *then* $D_1[R]$ *is bisimilar to* $D_2[R]$.[1]

Service nets are very similar to agent UML (AUML) interaction diagrams. Two examples are shown in Figures 2 and 3 (details concerning the content of these services are delayed until the following subsection). Each transition t of a service net D is assigned to the atomic role $r(t) \in Rol$ with the meaning, that only agents that implement this role r are able to execute the task t. Whenever a place p connects two transitions t_1 and t_2 with $r(t_1) = r(t_2)$ the transitions are drawn vertically on the so called *life line* of the role. Whenever $r(t_1) \neq r(t_2)$ then the place p models a *message exchange* which is drawn horizontally (for details cf. [11]). In Figure 2 all transitions drawn below one of the role names (*producer*, *consumer*) are assigned to it. The subnet $PC[producer]$ is indicated by the filled nodes.

2.2 R/D Nets and Formal Organizations

Based on the former specifications of roles and services we define R/D nets (role/delegation), which are used to describe formal organizations and embody both the structural and functional dimension of our model. R/D nets are Petri nets where each place models a role and each transition models a task. Each place p is labelled with a role $R(p)$ and each transition t with a service net $D(t)$ that describes how the corresponding task is to be carried out. For the net models it is a natural restriction to allow exactly one place in the preset of a task since each task is related to the role it can be used by.

Definition 3. *A* R/D net *is the tuple* (N, R, D) *where*

1. $N = (P, T, F)$ *is a Petri net with* $p^\bullet \neq \emptyset$ *for all* $p \in P$ *and* $|{}^\bullet t| = 1$ *for all* $t \in T$.
2. $\mathcal{R}$ *and* $\mathcal{D}$ *are a role universe and a service universe respectively.*
3. $R : P \to \mathcal{R}$ *is the role assignment.*
4. $D : T \to \mathcal{D}$ *is the service net assignment.*

An organizational structure in our model is built up by *organizational positions* (OP). OPs roughly correspond to a positions in real-world organizations. An OP may encompass several roles by being responsible for several tasks. In addition, it is possibly allowed to delegate role parts for the accomplishment of these tasks to other OPs. OPs are modelled as disjoint subsets of $P \cup T$ of an R/D net. We request the connectivity property. Whenever a task t belongs to an OP, then all used roles $p \in t^\bullet$ belong to it, whenever p belongs to an OP, so do all using tasks $t \in {}^\bullet p$.

[1] We omit a formal definition of bisimilarity. Its informal meaning in this context is that bismilar role components of service nets cannot be distinguished with respect to input/output behaviour.

Definition 4. *An* organization net *is a pair* $(N, \mathcal{O})$ *where* N *is a Petri net* $N = (P, T, F)$ *and* $\mathcal{O}$ *is a partitioning on the set* $P \cup T$ *where for all* $O \in \mathcal{O}$ *the following holds (with* $\bar{O} = (P \cup T) \setminus O$*):*

$$\forall p \in O \cap P : {}^\bullet p \subseteq O \wedge p^\bullet \subseteq \bar{O} \quad \wedge \quad \forall t \in O \cap T : {}^\bullet t \subseteq \bar{O} \wedge t^\bullet \subseteq O$$

An element $O \in \mathcal{O}$ *is called* organizational position *(OP). By* $O(y)$ *we denote the unique OP for each* $y \in P \cup T$*.*

A formal organization *is the tuple* $Org = (N, \mathcal{O}, R, D)$ *where* (N, R, D) *is a R/D net and* $(N, \mathcal{O})$ *is an organization net.*

Example 1. An organization net is given in Figure 1 (the purpose of the dashed lines will not be addressed until the following subsection).

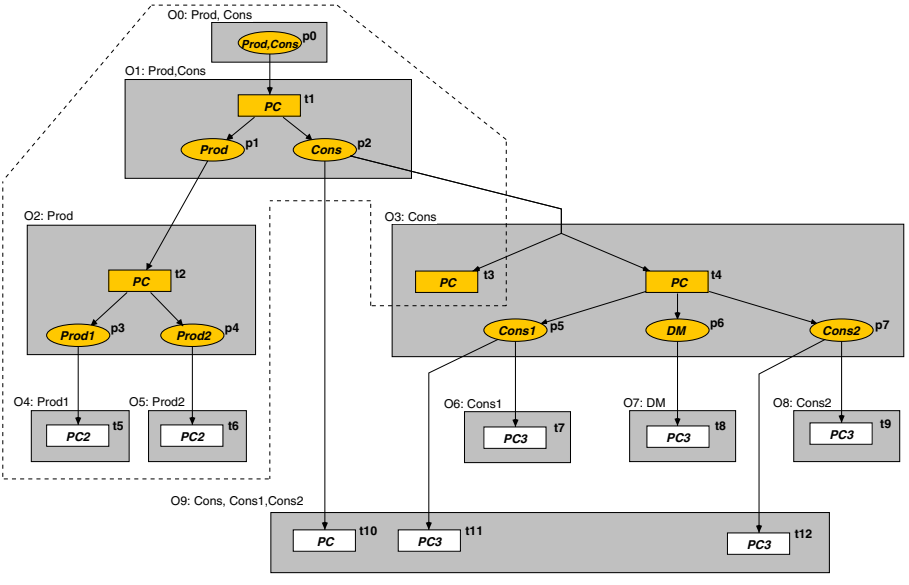


Fig. 1. Producer/Consumer organization net

OPs are depicted as filled rectangles. The OPs O_1, O_2, O_3, O_9 are complex, the OP O_0 is an initial one, and the other OPs are final ones. The organization net models a simple producer/consumer example. The place p_0 is the starting point of activity which has to start with $t_1 \in O_1$. In O_1 the work is divided into its producing and consuming parts, resulting in the roles *Producer* and *Consumer* respectively. The implementation of these roles is delegated to other positions. O_2 is the only OP to implement the role *Producer*. It does so by further dividing the work into two parts and delegating the resulting two roles $Producer_1$ and $Producer_2$ to the OPs O_4 and O_5 respectively, which finally implement their share of the work themselves. Turning to the consuming part of the work, OP

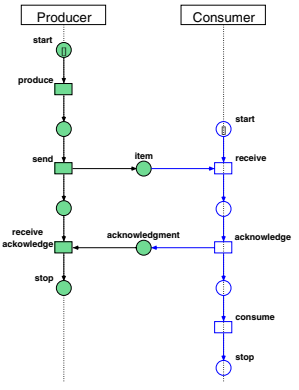


Fig. 2. Producer/Consumer: PC

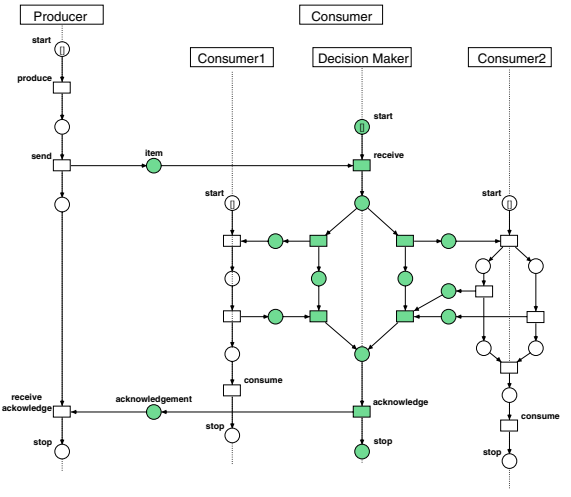


Fig. 3. Refined Producer/Consumer: PC_3

O_3 may decide to implement the role *Consumer* itself via task t_3 or to further divide the work and delegate the implementation of the resulting roles. Finally, OP O_9 is able to implement all variants of roles for consuming work. It might for example be considered an emergency OP for the case where the regular OPs responsible for consuming tasks are overloaded with work.

Formal models can be used to analyze structure and behaviour. As an example of this benefit to our approach we refer to *well-formedness* of R/D nets. In a well-formed R/D net, all service nets and all roles are consistently related to the structure of the organization net. We do not provide a formal definition here (cf. [9]) but give an example. Figure 2 shows the service net PC that is assigned to transition t_4 and describes the interaction between the two roles of *Producer* and *Consumer*. First the producer executes the transition *produce*, then *sends* the produced *item* to the consumer, who *receives* it. The consumer sends an *acknowledge* to the producer before he *consumes* the item. Figure 3 presents the service net PC_3 that is assigned to the transitions t_7, t_8 and t_9 and refines the role *Consumer* from the net PC of Figure 2: The role *Decision Maker* decides whether $Consumer_1$ or $Consumer_2$ receives the item. Basically, well-formedness in this case means that the refinement of the role *Consumer* into the roles $Consumer_1$, $Decision\ Maker$ and $Consumer_2$ that is modelled in the organization net is also mirrored in the corresponding service nets accordingly. The refinement concerns communication between the refining roles while input/output behaviour from the perspective of the producer remains the same.

2.3 Teams as Processes of Organization Nets

Organization nets capture all possibilities of task delegation and hence service accomplishment. In order to actually carry out organizational services, temporary

structures that are cleansed from ambiguities are required. We call these structures *teams* and formally define them as follows.

Definition 5. *A R/D net $N = (P, T, F)$ is called a* team *if N is causal net with exactly one minimal node: $^\circ N = \{p_0\}$, $p_0 \in P$, and all maximal nodes are transitions: $N^\circ \subseteq T$.*

In Figure 1 one possible team is framed by the dashed lines. This R/D net is dedicated to the specific purpose modelled by its only minimal node. All delegation ambiguities are resolved, resulting in a fixed allocation of task responsibilities to positions along with a fixed and fully elaborated (composed) interaction pattern for carrying out the corresponding organizational service. In this sense, the participating positions stand ready to act as a team.

In the following we characterize teams as the result of a team formation procedure that is controlled by the organizational structure. In particular, we relate teams to *processes* of organization nets. Petri net processes are a recognized alternative for describing the behaviour of Petri nets by firing sequences [12,13]. Processes are themselves Petri nets from the class of causal nets. A process of a net N is defined as a causal net K together with a pair of mappings $\phi = (\phi_P, \phi_T)$. For a mapping $f : D \to D'$, let $f^\sharp : \mathbb{N}^D \to \mathbb{N}^{D'}$ denote its extension to a homomorphism on multisets: $f^\sharp\,(\sum_{i=1}^n x_i) = \sum_{i=1}^n f(x_i)$.

Definition 6. *Let $N = (P, T, F)$ be a Petri net with the initial marking m, $K = (B, E, \lessdot)$ a causal net and $\phi = (\phi_P : B \to P, \phi_T : E \to T)$ a pair of mappings. Then (K, ϕ) is a* process *of (N, m) if the following holds:*

1. *Preservation of the flow relation: $x \lessdot y \Longrightarrow (\phi(x), \phi(y)) \in F$.*
2. *Representation of the initial marking: $\phi_P^\sharp(^\circ K) = m$.*
3. *Preservation of localities: ${\phi_P}^\sharp(^\bullet e) = {}^\bullet(\phi_T(e))$ and ${\phi_P}^\sharp(e^\bullet) = (\phi_T(e))^\bullet$.*
4. *Each node $x \in B \cup E$ has only finitely many predecessors.*

A process has the progress property *iff no transition is enabled in K°, i.e. for each subset $A \subseteq K^\circ$ there is no transition $t \in T$ such that $\phi(A) = {}^\bullet t$. The set of all finite processes with the progress property is denoted $\mathcal{K}(N, m)$.*

Alternatively, a process (K, ϕ) can be constructed from the possible firings, i.e. the enablement of transitions, of the net N. The construction is inductively defined for a process net, by adding transitions according to the enablement condition of the net N. The starting point is given by the initial marking, which defines a simple process without any transitions, but only a place for each token in the initial marking. For the progress property this unfolding is continued until no transition is enabled.

In [9] we prove a practical property: Each process with the progress property introduces a team.

Theorem 1. *Let $Org = (N, \mathcal{O})$ be an organization net with $N = (P, T, F)$. Then for each $p \in P$ and each process $(K, \phi) \in \mathcal{K}(N, \{p\})$ we have that K is a team.*

This theorem allows us to characterize team formation processes in terms of the theory of Petri net processes: Each Petri net process (with the progress property) of an organization net generates a team.

3 Multi-agent System Deployment

In this section we turn from formal organizations (without reference to members of the organization) as defined in Section 2 to their deployment inside multi-agent systems where agents populate an organization as its members. In order to benefit from the operational semantics of our Petri net-based approach, we require that the modelling of a formal organization does not just serve an offline design purpose but is directly operationalized within the software system and takes on steering and controlling responsibility. By this means, the formal specifications along with certain desired (provable) properties are directly mapped onto the resulting software system.

Consequently, our proposal specifically aims at multi-agent system architectures that allow for the direct deployment of Petri net models as software components. One particular candidate that most smoothly satisfies this requirement is the Mulan-architecture [14]. It models agents and multi-agent systems through the higher-order Petri net formalism of reference nets.[2] As reference nets may carry complex Java-instructions as inscriptions and thereby offer the possibility of Petri net-based programming, the Mulan models have been extended to a fully elaborated and running software architecture.

However, we consider our proposal not to be restricted to Mulan-deployment or to other target platforms that allow for the integration of Petri nets as executable software components. Instead, our Petri net models may be regarded as a special form of algorithmic description to be implemented in any desired way. In this case they still retain their character of being directly derived from the properties of the underlying Petri net-based organizational models.

3.1 Multi-agent System Design

In the following we regard a multi-agent system as an instance of a formal organization $Org = (N, \mathcal{O}, R, D)$. Each organizational position $O \in \mathcal{O}$ is allocated to an *organizational position agent* (OPA) $Ag(O)$. Figure 4 displays an example.

Position agents integrate the structural and behavioural organizational specifications associated with their respective positions into their local reasoning. Thus, organizational coordination plans and individual action plans are mutually related. The organizational coordination plans imprint the individual action plans and conversely the individual actions of the position agents produce and

[2] Reference nets show some extensions compared to conventional coloured Petri nets [15]. They implement the nets-within-nets paradigm where a surrounding net (the system net) can have *nets as tokens* (the object nets). Reference semantics is applied, so these tokens are *references* to *net instances*. *Synchronous channels* allow for communication between net instances.

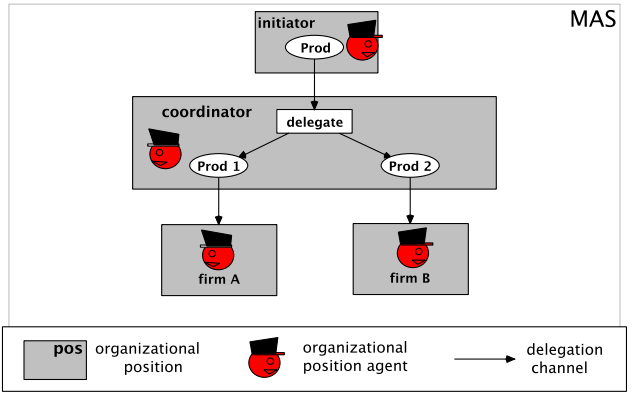


Fig. 4. Multi-agent system design derived from an organization net

reproduce the organization's global behaviour. In this sense we arrive at a dualistic relationship between micro and macro perspective within the multi-agent organization.

Position agents are connected through the delegation relationships of the underlying organization net $(N, \mathcal{O})$ with $N = (P, T, F)$. As $\mathcal{O}$ is a partition on the set $P \cup T$ each task $t \in T$ is assigned to exactly one position agent $(Ag \circ O)(t)$. If $t^\bullet = \emptyset$ holds, the corresponding position agent carries out the task itself. If $t^\bullet \neq \emptyset$, the corresponding position agent delegates the task. Two agents $Ag(O_1)$ and $Ag(O_2)$ are connected through a directed delegation relationship if $\exists p \in P : p \in O_1 \wedge p \in {}^\bullet O_2$ holds.

3.2 Operationalizing Organization Nets: The Team Formation Case

Besides guiding the design of corresponding multi-agent systems, organization nets may be directly operationalized in the implementation. Organizational processes are basically teamwork processes to carry out organizational services.[3]

Teamwork encompasses the stages of *team formation, team plan formation* and *team plan execution* [16]. The formal Petri net organizational models may guide all three stages and in the paper at hand we exemplarily elaborate on the case of team formation.

In our model, teams are formed through an iterated delegation procedure resting on the underlying organization net $(N, \mathcal{O})$ with $N = (P, T, F)$. Whenever a position agent $Ag(O)$ $(O \in \mathcal{O})$ has been chosen as a team member to implement the role belonging to a place $p \in {}^\bullet O \cap P$ it applies an *internal team formation function* to select one of the implementation possibilities from the set

$$Impl(p, O) = \{t \in T \cap O \mid t \in p^\bullet\} \ .$$

[3] In the context of this paper, we leave aside further aspects like for example processes for re-organization.

After choosing one element $t \in Impl(p, O)$ and if $t^\bullet \neq \emptyset$ holds the agent $Ag(O)$ further applies an *external team formation function* for all $p \in t^\bullet$. For each p, it selects one of the possible team members from the set

$$TM(p) = \{O \in \mathcal{O} \mid p \in {}^\bullet O\} \ .$$

This delegation process as a whole corresponds to firing sequences of the organization net and thus constructs the net's processes and hence teams.

The distributed team formation algorithm is given in pseudo-code in Figure 5. In accordance with Subsection 2.3, the algorithm generates a finite process with the progress property $(K, \phi) \in \mathcal{K}(N, \{p\})$ for $p \in P$ as the starting point. Distribution is denoted by the different selection functions τ^1 and τ^2 where $\tau^1_A(K, \phi, b) \in TM(\phi(b))$ denotes the team member that an agent A selects and $\tau^2_A(K, \phi, b) \in Impl(\phi(b), Ag^{-1}(A))$ denotes the task that an agent A selects.

4 A Petri-Net Model of Team Formation

The iterated delegation procedure for team formation according to the algorithm from Figure 5 is realized through a conversation between the involved position agents. We propose to utilize *deployment versions* of organization nets for each agent in order that they guide and control the conversation flow. Such a deployment version for a position agent basically takes the corresponding position fragment of the organization net and enriches it with inscriptions and additional net components for the operationalization. Deployment versions need not be modelled manually but can be generated automatically as all information that is needed rests in the initial corresponding organization net.

For the MULAN-architecture this approach is straightforward. In [14], it is described in detail how MULAN-agents participate in conversations. Each agents hosts an instantiated *protocol net* that implements the agent's role in the conversation by receiving and sending messages along a predefined pattern of communication. So deployment versions of organization nets may be integrated in

---

function $K_\tau(N, p)$ is
  $(K, \phi) := ((\{b_0\}, \emptyset, \emptyset), \{(b_0, p)\})$
  while $(K^\circ \cap B) \neq \emptyset$ do
    foreach $b \in (K^\circ \cap B)$ do
      $(K, \phi) := ((B', E', F'), \phi')$ where
        $TM = \tau^1_A(K, \phi, b)$ where $A = (Ag \circ O \circ \phi)(b)$
        $t = \tau^2_{TM}(K, \phi, b)$
        $B'_t = \{b_p \mid p \in t^\bullet\}$; $\phi' = \phi \cup \{(e, t)\} \cup \{(b_p, p) \mid p \in t^\bullet\}$
        $B' = B_K \uplus B'_t$; $E' = E_K \uplus \{e\}$; $F' = F_K \cup \{(b, e)\} \cup (\{e\} \times B'_t)$
  return (K, ϕ)

Fig. 5. The Team Formation Algorithm

protocol nets as object net tokens in order that they guide and control the conversation. For multi-agent system architectures other than MULAN the integration is likely to be more complicated. However, it does not differ *conceptually* as defining agent interaction patterns and distributing parts of them among agents according to different participating roles is common practice.

4.1 Team Formation: Position-Specific Deployment Version

Deployment fragments of organization nets for the case of team formation will now be examined in more detail. To begin with, we refer to our example from Section 2.2 illustrated in Figure 1. The deployed organization net for the position agent assigned to OP O_3 is shown in Figure 6.[4] It contains the subnet for O_3 as a substructure.

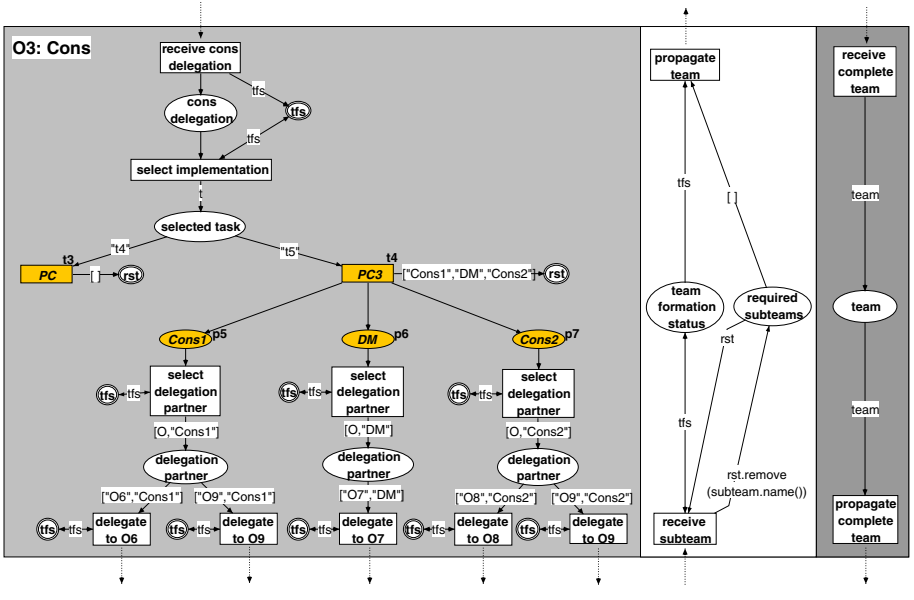


Fig. 6. Team formation: Deployment version for OP O_3 from Figure 1

Upon receiving the delegation call to implement the role *Consumer* (receive cons delegation), the team formation status is initialized (team formation status).[5] The team formation status encapsulates information about delegation

[4] To avoid overloading the figure with too many details it does not contain all inscriptions that would be necessary for operationalization. This holds especially for the transitions whose function is only given by their name.

[5] The twofold circuited places are *virtual places*. Reference nets offer virtual places as exact copies of the original ones for the sake of readability, avoiding arcs all across the diagram. In this case the virtual places tfs are copies of the place team formation status and the virtual places rst are copies of the place required subteams.

partners (both higher and lower in delegation authority), team formation parameters (requirements, constraints), selected tasks and recursively generated subteams. The team formation status is updated when one of the two possible implementations for *Consumer* is chosen (select implementation). Depending on the selected task (t_3 or t_4), different subteams are required (required subteams). If the position agent decides to implement the role itself (t_3) no further subteams are required. If the position agent decides to further delegate the implementation of the task (t_4) the work is divided into further roles for delegation ($Consumer_1$, $DecisionMaker$, $Consumer_2$) and each of these requires its own subteam. Delegation partners have to be selected among possibly multiple options (select delegation partner) and the team formation status has to be updated accordingly. Eventually, the roles are delegated to the selected team members (delegate to O_x).

Whenever a subteam for a formerly delegated role is received (receive subteam), the team formation status is updated and the subteam is removed from the list of required subteams. If this list is empty, all required subteams have been received. Now the team formation status contains all information that is required to generate the team for the delegation level of O_3 and can be forwarded to the higher level of delegation authority (in this case O_1) as an implementation for the role *Consumer* (propagate team).

Finally, the position agent will receive the complete team description from its higher level of delegation authority (receive complete team) and forwards it to its lower level team members (propagate complete team).

Recall that deployment versions of organization nets are to be utilized inside protocol nets (or some other mechanism to implement agent participation in conversations). It is not at the level of the deployed organization net that issues like mapping of agents to positions, addressing or message generation are dealt with. These are deferred to the next higher level of conversation management. Deployed organization nets only contain information that can directly by derived from the underlying organization net.

It should be obvious that the transitions select implementation and select delegation partner are the most prominent options for refinement. Different scenarios are imaginable. For example a position agent might have to accept all delegations at all odds. Otherwise it might be allowed to negotiate performance conditions depending on its own workload and that of its lower delegation partners. Or it might even be allowed to reject delegation calls.

4.2 Team Formation: General Deployment Version

The deployed organization net from Figure 6 is specifically tailored for OP O_3 from Figure 1. It is a very simple example with just one partner of higher delegation authority and only two implementation variants. Still, the deployment version is a quite large net due to operationalization overhead. More complex organizational positions would lead to very large and probably hard to arrange

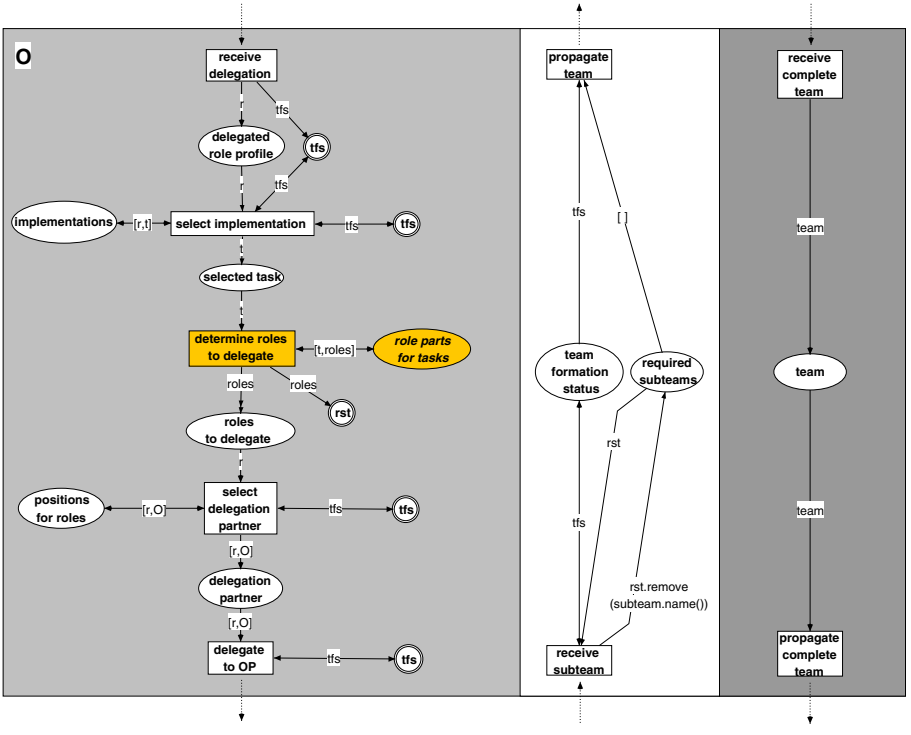


Fig. 7. Team formation: General deployment version for organizational positions

deployment versions. Instead, Figure 7 shows a coloured version of the former deployment version that can be used for an *arbitrary* organizational position.

Basically, the generalized version introduces associations between delegated roles and tasks (implementations), tasks and roles to delegate (role parts for tasks), and roles and organizational positions (positions for roles) as side conditions for the selection of implementations, the determination of role parts for an implementation and the selection of delegation partners respectively. Each association is simply a tuple that rests as a token on the corresponding place. Consequently, an organizational position might be arbitrarily complex and still be tackled by the deployment version from Figure 7 for the sake of team formation. A rise in complexity does only result in additional tokens on the places implementations, role parts for tasks and positions for roles.

This deployment version is still only usable for one occurrence of a delegation call. If the tokens related to specific delegations would be arranged as tuples with an identification number as a further element, the net from Figure 7 could even be used as a permanent component that all protocol nets (or some other mechanism to implement agent participation in conversations) could use jointly and simultaneously.

5 Outlook

We presented an approach to use Petri nets to both specify and deploy organizational models. The result is an integrated approach to develop organization-oriented software systems that especially focuses on closing the conceptual gap between design and implementation. Moreover, the approach explicitly facilitates change. As the specification is immediately used to structure and operationalize the multi-agent system and deployment versions of organization nets can be generated at run time, changing the specifications consequently changes the structure and behaviour of the software system.

Turning to future work, we put a strong focus on expanding the software technical framework that supports multi-agent organizations according to our mathematical model. In particular, our approach follows the *common organization implementation architecture for open multi-agent systems* introduced in [2]. This approach distinguishes between the organizational and the domain layer of applications. Our interpretation of this concept is illustrated in Figure 8. We distinguish the already mentioned organizational position agents (OPA) from organizational member agents (OMA) (cf. also [17]). The organization contracts each position to a member agent that carries out actions and makes decisions. But member agents must connect to the corresponding position agents and use them as (controlling and coordinating) gateways to the organization. Thus, the position agents embody a distributed middleware that constrains the member agents' impact on the organization while at the same time supporting them by providing organizational services (more specifically, automated support for team formation, team plan formation and teamwork execution).

This middleware approach is particularly helpful in the case of an open multi-agent system environment with agents belonging to different stakeholders entering and leaving on a continual basis. It also aims at incorporating our model into existing multi-agent systems. No matter what the predominant structures and relationships are, our model can introduce a new organizational level (or just selected organizational features/domains) by means of a position agent infrastructure to which existing agents connect as members and which they use for coordination. The former structures and relationships of course do not become obsolete but correspond to the informal channels besides the formal ones defined by the organization.

A further valuable prospect of the OPA/OMA distinction opens up with respect to hierarchical nesting of whole *organizational units*. In this paper we defined the conceptual and operational connection of positions to their organizational embedding. As one next step we want to regard a position as possibly being occupied by a complex organizational unit in itself like it is shown in Figure 8. This step facilitates a more modular conception of large organizational settings where the need arises to distinguish between different system levels and to abstract away from lower-level issues at each level respectively. We have presented a particular proposal for a multi-level architecture following this conception of nested organizational units in [1,18]. Four levels of organizational units are distinguished according to different degrees of abstraction, namely *department,*

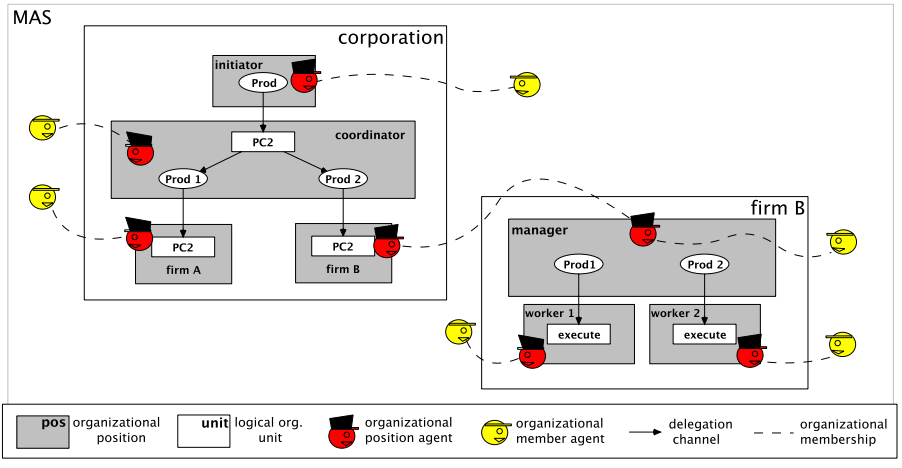


Fig. 8. Nested multi-agent system deployment

organization, organizational field and *society*. These levels are analogous to the levels of analysis that Scott proposes in [8] for organizations in human societies. We consider this architecture to accompany a shift in paradigm from agent-oriented to organization-oriented software engineering. It is no longer the agent that represents the central design metaphor and software building block but the organization.

References

1. Wester-Ebbinghaus, M., Moldt, D., Reese, C., Markwardt, K.: Towards Organization–Oriented Software Engineering. In: Züllighoven, H. (ed.) Software Engineering Konferenz 2007 in Hamburg: SE 2007 Proceedings. LNAI, vol. 105, pp. 205–217. GI (2007)
2. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open organizations. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS, vol. 4457, pp. 86–105. Springer, Heidelberg (2007)
3. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional and deontic specification of organizations in multiagent systems. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS, vol. 2507, pp. 439–448. Springer, Heidelberg (2002)
4. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS, vol. 2333, pp. 348–366. Springer, Heidelberg (2002)
5. Pynadath, D., Tambe, M.: An automated teamwork infrastructure for heterogeneous software agents and humans. Autonomous Agents and Multi-Agent Systems 7(1-2), 71–100 (2003)
6. Sonenberg, E., Tidhar, G., Werner, E., Kinny, D., Ljungberg, M., Rao, A.: Planned team activity. In: Castelfranchi, C., Werner, E. (eds.) MAAMAW 1992. LNCS, vol. 830, pp. 227–256. Springer, Heidelberg (1994)

7. Mintzberg, H.: Structure in Fives: Designing Effective Organizations. Prentice-Hall, Englewood Cliffs (1983)
8. Scott, W.R.: Organizations: Rational, Natural and Open Systems. Prentice-Hall, Englewood Cliffs (2003)
9. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae 79(3-4), 415–430 (2006)
10. Girault, C., Valk, R.: Petri nets for systems engineering: a guide to modelling, verification and applications. Springer, Heidelberg (2003)
11. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring Petri net-based agent interaction protocols. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 102–120. Springer, Heidelberg (2003)
12. Goltz, U., Reisig, W.: The non-sequential behaviour of Petri nets. Information and Control 57, 125–147 (1983)
13. Best, E., Fernández, C.: Nonsequential processes: a Petri net view. Springer, Heidelberg (1988)
14. Köhler, M., Moldt, D., Rölke, H.: Modelling the structure and behaviour of Petri net agents. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 224–241. Springer, Heidelberg (2001)
15. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
16. Wooldridge, M., Jennings, N.: The cooperative problem-solving process. Journal of Logic and Computation 9(4), 563–592 (1999)
17. Köhler, M., Wester-Ebbinghaus, M.: Petri net-based specification and deployment of organizational models. In: Moldt, D., Kordon, F., van Hee, K., Colom, J.M., Bastide, R. (eds.) Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE 2007), Siedlce, Poland, Akademia Podlaska, pp. 67–81 (2007)
18. Wester-Ebbinghaus, M., Moldt, D.: Structure in threes: Modelling organization-oriented software architectures built upon multi-agent systems. In: Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008), pp. 1307–1311 (2008)

Exploring Robustness in the Context of Organizational Self-design

Sachin Kamboj and Keith S. Decker

Department of Computer and Information Sciences
101 Smith Hall
University of Delaware
Newark, DE 19716
{kamboj,decker}@cis.udel.edu

Abstract. Robustness is the ability of a multiagent system to recover from failures and exceptions. In particular, the system should be able to recover from task and agent failures and the failure of any single agent or group of agents should allow the graceful, predictable degradation of performance. Hence, designing robust systems is a critical challenge facing many multiagent designers.

Organizational Self-Design (OSD) has been proposed as an approach to constructing suitable organizations at runtime in which the agents are responsible for constructing their own organizational structures. OSD has also been shown to be especially suited for environments that are dynamic and semi-dynamic. However, the problem of making these self-designed organizations robust is still an open research problem that has not been studied to any considerable extent. In this paper, we focus on developing and evaluating robustness mechanisms that can be used by the agents in conjunction with OSD.

1 Introduction

Robustness may be defined as the ability of a multiagent system to recover from failures and exceptions. An exception may be defined as a departure from an "ideal" system behavior [1]. Recovery would then involve the execution of some corrective measures to reinstate the ideal system behavior.

Achieving robustness is particularly challenging in dynamic and semi-dynamic environments, since the problem characteristics, available resources or agent capabilities may change over time. Multiagent organizations for such environments must include two components — the first component is responsible for monitoring the performance of the organization and for discerning whether or not the measured performance falls within the design parameters of the organization. The second component is responsible for explicitly changing the organization if it fails to meet its design goals.

This monitoring and design may be done by an entity external to the organization (i.e. by the multiagent designer) or by the constituent agents themselves. In an extreme form of the latter approach, the agents come up with a new, implicit, one-off organization for each new problem instance. This is what happens in the contract net protocol (CNP) [2].

This latter approach is referred to in the literature as organizational self-design (OSD) [3,4] and in this approach the agents are responsible for designing their own

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 80–95, 2009.

organizational structure at run-time. Any OSD approach needs to provide answers to questions like how many agents are needed, how are the tasks and resources divided amongst the agents and what coordination mechanisms are suitable for the problem at hand. OSD is especially suited in situations where the environment is semi-dynamic as the agents can adapt to changes in the task structures and environmental conditions, while still being able to generate relatively stable organizational structures that exploit the common characteristics across problem instances.

One application of OSD is to allocate resources in grid/cloud/volunteer computing systems. Volunteer computing [5,6] is a form of distributed computing in which a group of volunteers donate their computing resources to a cause, such as folding proteins, predicting climate change, etc. In order to do this, the volunteers have to download a client, which then connects to one or more centralized servers and requests jobs that make use of the volunteer's computing resources. The jobs are then executed on the volunteers' machines and the results are sent back to the servers. The centralized servers, in turn, need to figure out a scheduling policy that tries to perform an optimal allocation of jobs to the clients (for some definition of optimality).

The clients running on the volunteer machines can be thought of as agents. This leads to a direct mapping from the problem of determining a suitable scheduling policy for the clients to the problem of determining a suitable organization for the agents. Hence, the solution to the organizational issues, such as the allocation of agents to the subtasks of the problem being solved and the coordination of inter-agent activities, will generate a scheduling policy that can be used to allocate jobs to the agents.

One of the consequences of using volunteer computing is that the availability of any volunteer client (i.e. agent) cannot be guaranteed as (a) volunteers are free to leave at any time without any warning or penalty; and (b) the volunteers' computers might crash or be switched off. We define the non-availability of a volunteer client as an *agent failure*. Hence, if OSD is to be used to allocate tasks and resources in a volunteer computing environment, it should be able to survive agent failures.

In this paper we would like to study various approaches that can be used to increase the robustness of organizations generated through the use of OSD. Furthermore, we primarily focus on agent failures[1] in worth-oriented domains. Our approach is geared towards real-world applications in grid/volunteer/cloud computing where a large pool of agents is available though the availability of any single agent cannot be guaranteed.

A primary challenge to incorporating robustness in the OSD process is the continuously changing set of agents and the roles that they enact. This leads to a continuously changing (and distributed) organizational knowledge that must be preserved across agent failures. Furthermore, one of the principal reasons for using OSD is the amortization of organizational costs across problem instances. This amortization might come at a price — such generated organizations might be particularly susceptible to agent failure. This is because the constituent agents are responsible for enacting particular

[1] Another aspect to robustness in multiagent systems is task failures. We won't be concerned with task failures because — (a) agent failure is significantly harder than task failure because the failure of an agent results in a loss of both its organizational knowledge and its contextual problem-solving state; and (b) our approach can easily handle task failures through rescheduling[7].

goals within the organization; and the failure of a single agent, performing a critical role, could bring down the whole organization. Hence, such organizations might be *less* robust than the one-off organizational schemes (such as CNP).

We feel that robustness has not been studied in the context of OSD to any significant extent. Whereas we [4] alluded to this problem in our previous paper, we did not present any algorithms and did not discuss the problem in any depth. The algorithm presented in [8] would respond to a single agent failure by performing a complete reorganization — an extremely expensive process. [9] also address robustness by reorganizing, however their approach is specific to the distributed sensor network and would have to be adapted for general purpose worth oriented domains. Other approaches to OSD [3,10,11] tend to completely skim over the problem.

This paper will present and evaluate algorithms for both of the two commonly used approaches to robustness:

1. the *Citizen Approach* [1,12], which involves the use of special monitoring agents (called *Sentinel Agents*) in order to detect agent failure and dynamically startup new agents in lieu of the failed ones.
2. the *Survivalist Approach* [13], which involves using the domain agents to monitor themselves. The domain agents may (a) restart failed agents and (b) create additional replica agents which may take over should the original agents fail.

Our goal is to allow the organization, at its best, to function without any performance degradation in the face of failures. At its worse, the organization should degrade gracefully in proportion to the number of failures.

Note that we do *not* present any new approach to OSD in this paper. Instead we add robustness to the OSD approach presented in [4]. Also, we are *not* trying to develop any new and general approaches to robustness. Instead we were trying to address the robustness issues that arise when using OSD. Our primary contribution is an analysis of the different approaches to robustness when using OSD to design organizations, so that a multiagent user can select the most suitable approach for their application.

The organization of the rest of this paper is as follows. In the next section we discuss the task model that we use for worth-oriented domains. This is followed by a discussion of our approach to OSD. Finally, we evaluate the presented algorithms.

2 Task Model

We use TÆMS as the underlying representation for our tasks (problem instances). TÆMS [14] (Task Analysis, Environment Modeling and Simulation) is a computational framework for representing and reasoning about complex task environments in which tasks (problems) are represented using extended hierarchical task structures [15]. The root node of the task structure represents the high-level goal that the agent is trying to achieve. The sub-nodes of a node represent the subtasks and methods that make up the high-level task. The leaf nodes are at the lowest level of abstraction and represent executable methods – the primitive actions that the agents can perform. The executable methods, themselves, may have multiple outcomes, with different probabilities and different characteristics such as quality, cost and duration. TÆMS also allows various

mechanisms for specifying subtask variations and alternatives, i.e. each node in TÆMS is labeled with a characteristic accumulation function that describes how many or which subgoals or sets of subgoals need to be achieved in order to achieve a particular higher-level goal. TÆMS has been used to model many different problem-solving environments including distributed sensor networks, information gathering, hospital scheduling, EMS, and military planning. [16,17,15,18].

For a more formal description of our task and resource model, please refer to [4,16].

3 Organizational Self Design

In our approach, problem solving requests arrive at the organization continuously at varying rates and with varying deadlines. To gain utility, the agents in the organization need to solve the problems by the given deadlines. The organizational design is directly contingent on the task structure of the problems being solved and the environmental conditions under which the problems need to be solved. Here, the environmental conditions refer to such attributes as the task arrival rate, the task deadlines and the available resources. We assume that all problems have the same underlying task structure, henceforth called *the global task structure*.

To participate in the organization, each agent must maintain some organizational knowledge. This knowledge is also represented using TÆMS task structures, called the local task structures. These local task structures are obtained by rewriting the global task structure and represent the local task view of the agent vis-a-vis its role in the organization and its relationship to other agents. Hence, all reorganization involves rewriting of the global task structure. However, note that the global task structure is *NOT* stored in any one agent, i.e. no single agent has a global view of the complete organization. Instead each agent's organizational knowledge is limited to the tasks that it must perform and the other agents that it must coordinate with — it is this information that is represented using the local task structures.[2]

To allow the agents to store information about other agents in the task structure, we augment the basic TÆMS task representation language by adding organizational nodes. To differentiate organizational nodes from "regular" TÆMS nodes, we refer to non-organizational nodes as *domain* nodes. The four organizational nodes are:

1. **Non-Local-Nodes** are used to represent a domain node in some *other* agent's local task structure. Non-Local-Nodes are used to represent nodes in the global task structure that the agent knows the identity (label) of but does *not* know the characteristics (e.g. quality, cost duration) of[3].
2. **Container-Nodes** are aggregates of domain nodes and other organizational nodes. Container nodes also have a type which determines their purpose. For the purposes

[2] Note that rewriting the local task structure results in a change in the goals and commitments of an agent. After every rewrite of a task structure, the agents that change usually renegotiate the coordination mechanism used to coordinate between themselves. The exact details are beyond the scope of this paper.

[3] At least initially at the time of breakup. It can however learn these characteristics through some coordination mechanism.

of this paper, there are only two types of container nodes: (a) *ROOT* nodes are used to store the high-level goals of an agent and the non-local nodes; and (b) *CLONE* nodes are used to store cloned portions of a TÆMS subtree.

3. **Clone Selectors** are used to select amongst the clones of a node. The purpose of a selector node within a clone-container is to *enable* one or more of the clones, so that the enabled nodes can be "executed" by the agents owning those clones.
4. **NLE-Inheritors** are methods whose sole purpose is to transfer the non-local effect from a non-cloned node to a cloned node or vice versa.

Algorithm 1. CHANGEORGANIZATION

```
1. if ISAGENTOVERLOADED() then
2.     SPAWNAGENT()
3. else if ISAGENTUNDERLOADED() then
4.     COMPOSEAGENT()
5. end if
```

To allow for a change in an agent's organizational knowledge, we define three rewriting operations on a task structure:

Breakup: Breakup involves dividing the local task structure of an agent so that it can be allocated to another agent. When a spawning agent divides a local task structure, A into two subparts B (for itself) and C (for the spawned agent), it still needs to maintain some knowledge about the tasks/methods in C while, at the same time, allowing the spawned agent to have as much autonomy as possible about the execution of C. Specifically the agent will need to know about the subset of nodes in C that are interrelated to the nodes in B, either through NLEs or through subtask relations. These interrelated nodes are represented using Non-Local-Nodes that have the same label as the domain nodes and correspond to the domain nodes. The algorithm for breakup is shown in Algo. 2 and an example is shown in Figure 1. In this example, the task structure represented by Root-1 is broken twice — once at node D to generate Root 2 and a third time at Node E to generate Root 3.

Merging: Merging involves combining two different local task structures from two different agents to form one local task structure. The algorithm for merging is shown in Algo. 3 and an example is shown in Figure 1. The core of the merging algorithm (lines 3 –12) involves finding "overlapping" nodes, i.e. identical nodes that are represented in both the task structures being merged. The merge algorithm goes through all the nodes in the second task structure and tries to find the corresponding nodes in the first task structure. If a corresponding node is found, the algorithm decides which of the two nodes should be kept and which of the nodes should be discarded.

Cloning: Cloning involves creating a complete copy of a substructure, so that it can be allocated to another agent. Cloning serves two purposes: (a) it can be used for load balancing similar to the work of [19]; and (b) it can be used to increase the *robustness capacity* of an agent by having multiple agents work on the same task simultaneously.

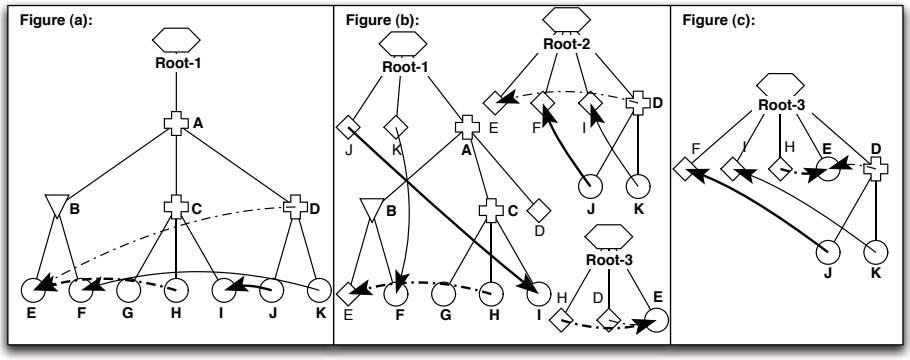


Fig. 1. Task rewriting: *Fig (a)* shows the global TÆMS task structure: The polygons (labelled A – D) represent tasks and the circles (labelled E – K) represent executable methods. The + iconography indicates a *SUM* CAF while $\bigtriangledown$ represents a *MIN* CAF. The arrows represent NLEs — the thick arrows represent hard constraints such as *Enables* (represented by a solid arrow from J to I) and *Disables* (represented by a broken arrow from H to E). The thin arrows show soft constraints such as *Facilitates* (solid arrow from K to F) and *Hinders* (broken arrow from D to E). Method characteristics and other details are omitted. *Fig (b)* shows the breakup of Root-1 at nodes D and E. The diamonds represent non-local nodes, that are the responsibility of some other agent. *Fig (c)* shows the merging of Root-3 and Root 2.

Algorithm 2. BREAKUP (τ, υ)

1. $\overline{\tau} \Leftarrow$ DESCENDENTS(τ) − DESCENDENTS(υ)
2. $\overline{\upsilon} \Leftarrow$ DESCENDENTS(υ)
3. **for all** $\{ N \mid N \in$ NLEs$(\tau) \}$ **do**
4. **if** (SOURCE$(N) \in \overline{\tau}$ **and** SINK$(N) \in \overline{\upsilon}$) **or** (SOURCE$(N) \in \overline{\upsilon}$ **and** SINK$(N) \in \overline{\tau}$) **then**
5. $x \Leftarrow$ GETNONLOCALNODE(SOURCE(N))
6. $y \Leftarrow$ GETNONLOCALNODE(SINK(N))
7. $M \Leftarrow$ COPYNLE(N)
8. REPLACENODE$(N$, SOURCE$(N), x)$
9. REPLACENODE$(M$, SINK$(N), y)$
10. **end if**
11. **end for**
12. $x \Leftarrow$ GETNONLOCALNODE(υ)
13. REPLACENODE(τ, υ, x)
14. **return** CreateRootNode(υ)

The algorithm for cloning is given in Algo. 4 and an example is shown in Figure 2. Lines 4–7 of this algorithm involve creating a copy of all the nodes in the subtree being cloned. Lines 8–21 are used to take care of NLEs in the cloned subtree that have a source or destination as a non-clone node. Such NLEs that transcend clone boundaries have to handled carefully in order to (a) preserve their original semantics and (b) allow the presence of clones to be transparent to the

Algorithm 3. MERGE (τ, υ)

```
1. for all { y | y ∈ DESCENDENTS(υ) } do
2.    x ⇐ FINDNODE(τ, x)
3.    if NULL(x) then
4.       DELETENODE(υ, y)
5.       ADDNODE(τ, y)
6.    else if TYPE(τ, x) = NonLocal and TYPE(υ, y) = NonLocal then
7.       MERGENODES(τ, x, y)
8.    else if TYPE(τ, x) = Local and TYPE(υ, y) = NonLocal then
9.       DELETENODE(υ, y)
10.   else if TYPE(τ, x) = NonLocal and TYPE(υ, y) = Local then
11.      REPLACENODE(τ, x, y)
12.   end if
13. end for
14. return τ
```

Algorithm 4. CLONE (τ, υ)

```
1. τ̄ ⇐ DESCENDENTS(τ) − DESCENDENTS(υ)
2. ῡ ⇐ DESCENDENTS(υ)
3. φ ⇐ CREATECLONECONTAINER(υ)
4. for all { x | x ∈ ῡ } do
5.    y ⇐ COPYNODE(x)
6.    ADDNODE(φ, y)
7. end for
8. for all { N | N ∈ NLEs(υ) } do
9.    if SOURCE(N) ∈ τ̄ then
10.      x ⇐ CREATEINHERITINGNODE()
11.      ADDNODE(φ, x)
12.      L ⇐ COPYNLE(N)
13.      M ⇐ COPYNLE(N)
14.      REPLACENODE(N, SINK(N), x)
15.      REPLACENODE(L, SOURCE(L), x)
16.      REPLACENODE(M, SOURCE(M), x)
17.      y ⇐ FINDNODE(φ, SINK(M))
18.      REPLACENODE(M, SINK(M), y)
19.   else if SINK(N) ∈ τ̄ then
20.      {Similar to the source}
21.   end if
22. end for
23. ADDNODE(φ, υ)
24. return φ
```

non clone nodes. In order to achieve this effect, we create special methods called *NLE-Inheritors*. These methods are simply conduits for the effects from the cloned nodes to the non-clone nodes.

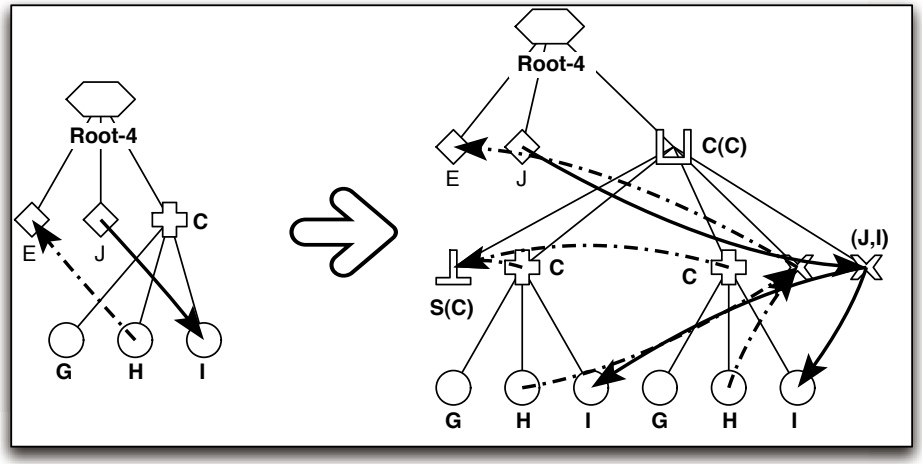


Fig. 2. Task Rewriting (cont.): *Figure showing the cloning of Node C in Root 4*. The ⊔ node (Node C(C)) is used to represent a clone container, a node created for storing the "clones" of a node; the ⊥ node (Node S(C)) is used to select which clone to "execute" — for the purposes of robustness, all the clones will be executed; finally the × nodes represent the NLE-inheriting-methods — these nodes form a conduit for NLEs to the non-clone parts of the task structure.

Our approach to OSD involves starting with a single agent responsible for the global task structure (i.e. the local task structure is equal to the global task structure). Each agent in the organization follows the algorithm presented in 1:

- If an agent is overloaded it either breaks or clones its local task view and then spawns a new agent. We define an overloaded agent as one that cannot complete the tasks in its task queue by their given deadlines.
- If the agent is underloaded, on the other hand, it composes with another agent, merging the local task structures of the two organizations. We define an underloaded agent as one that is idle for an extended period of time.

For more details on the precise mechanism used to detect overload and underload please refer to [4].

4 Robustness Mechanisms

Both of our robustness mechanisms involve three parts: (a) monitoring for agent failure; (b) maintaing state information for all the agents; and (c) restarting failed agents.

Furthermore, the underlying mechanism for monitoring and restarting is the same across the robustness mechanisms. Monitoring is achieved by sending out periodic *Are-You-Alive* messages to the set of monitored agents and waiting for *Alive* reply messages. If a reply is not received within a certain interval, we assume that the agent is dead and send a restart message to the environment. The individual mechanisms, however, differ in *who* is responsible for the monitoring and *which* set of agents are monitored.

State information is needed to restart a failed agent. At a minimum, this state information should contain the *organizational state* (i.e. the local task structure) of the agent being restarted. However, the local task information is not sufficient for restarting an agent in a complex domain. The restarted agent will still need information about the *execution context*, i.e. information about the outstanding task instances, information about the methods of a task instance that have already been executed (so that the agent does not try to re-execute them) and information about coordination commitments (because the subtasks have non-local effects and are interdependent on each other). The coordination mechanisms also differ in how they keep a track of this execution context.

4.1 Citizens Approach

The citizens approach involves creating a special monitoring agent (called a *sentinel agent*), which is responsible for all the robustness related responsibilities of the organization. This approach is the simplest to execute — the sentinel agent is the sole monitor that is responsible for monitoring all the agents in the organization. However, to maintain state information, the sentinel needs to listen to *all* the messages exchanged by *all* the other agents, since it needs to store both the set of spawning/composition messages (in order to track the changes to the local task structures of the agents) and the set of execution/coordination messages between the agents (in order to keep a track of the execution context).

Hence, not only does the sentinel effectively becomes a conduit for all the messages, it also has global knowledge about the complete organization — a problem we were trying to avoid by using the OSD approach in the first place. Furthermore, the sentinel can (a) quickly become overwhelmed by all the messages that it needs to track and (b) become a central point of failure[4]. The solution might be to add multiple sentinel agents — we will now need to create an organization for the sentinels (for which we could, again, use OSD) and a way of monitoring the monitors.

Hence, we focus on developing algorithms for the survivalist approach and use the citizens approach for comparison.

4.2 Survivalist Approach

In the survivalist approach, there are no special agents responsible for monitoring and restarting failed agents. Instead the domain agents divide the monitoring responsibilities amongst themselves. Furthermore, some/all domain agents may be replicated in order to (a) increase the robustness capacity of the organization; (b) decrease the response time to a failure, and (c) process task instances in parallel, thus helping to balance the load.

The obvious advantage of the survivalist approach is that no one agent is overburdened with the monitoring responsibilities. Also there is no central point of failure and no agent with global knowledge of the organization. Furthermore, the survivalist approach can take into account the interplay between a satisficing organizational structure and probability of failure. For example, one way of achieving a higher level of robustness in the survivalist approach, given a large numbers of agent failures, would be to

[4] It's unreasonable to assume that the other agents might fail, but the sentinel will never fail.

relax the task deadlines. However, such a relaxation would result in the system using fewer agents in order to conserve resources, which in turn would have a detrimental effect on the robustness. These advantages come at a cost of increased complexity of the monitoring mechanism.

Creating a monitoring set of agents. The monitoring set of an agent, *Agent A*, is defined as the set of agents that are responsible for monitoring *Agent A* for failures. We assume that the minimum cardinality of this set, *N* is an input to the organization[5]. Also in our approach, all monitoring is mutual, i.e. if *Agent A* is in the monitoring set of *Agent B* (i.e. if *Agent A* is responsible for monitoring the health of *Agent B*), then *Agent B* is in the monitoring set of *Agent A*. This is by design, because *Agent A* on receiving an *are-you-alive* request from *Agent B*, already knows that *Agent B* is alive and does not need to send *Agent B* a separate request.

Each agent is responsible for determining its monitoring set. At the time an agent, say *Agent A*, is first spawned, it runs the following algorithm:

1. *Agent A* determines its related set. The related set of *Agent A* is the set of agents that have a coordination relationship with *Agent A*. (This coordination relationship would exists because of interdependent tasks and NLEs in the task structures of the agents).
2. If the number of agents in the related set of *Agent A* is greater than N, *Agent A* sends a message to each of the related agents, requesting the cardinality of their respective monitoring sets, and then goes to Step 3. If this number is less than N, *Agent A* adds all of the related agents to its monitoring set and then jumps to Step 4.
3. *Agent A* picks the N related agents with the lowest monitoring-set cardinalities to be in its monitoring set.
4. *Agent A* sends messages to all the other non-related agents, requesting their monitoring-set cardinalities. (This can be done using a single broadcast message). *Agent A* then iteratively selects agents with the lowest monitoring-set cardinalities until either (a) it has N agents in its monitoring set or (b) until all the agents have been exhausted (i.e. there are less than N agents in the whole organization).

Finally, once *Agent A* has determined its monitoring set, it can send a message to each of the agents in its set requesting them to monitor its health. In addition to being distributed, other advantages of this algorithm are: (a) Steps 1–3, can be piggy-backed onto the coordination-mechanism negotiation messages exchanged with the agents in the related set and (b) This scheme will reduce the frequency of *are-you-alive* messages transmitted since the agents will be communicating in-band as a part of their normal tasks processing.

Augmenting the robustness capacity of an organization. The robustness capacity of an organization is defined as the number of agent failures that an organization can withstand. The robustness capacity is equal to the kill count minus 1, where the kill count is the *minimum* number of agent that need to be killed in order to kill the organization

[5] It should be possible to develop an algorithm for learning the optimal value of N given the environment conditions — i.e. the probability of failure. We plan to incorporate this into our future work.

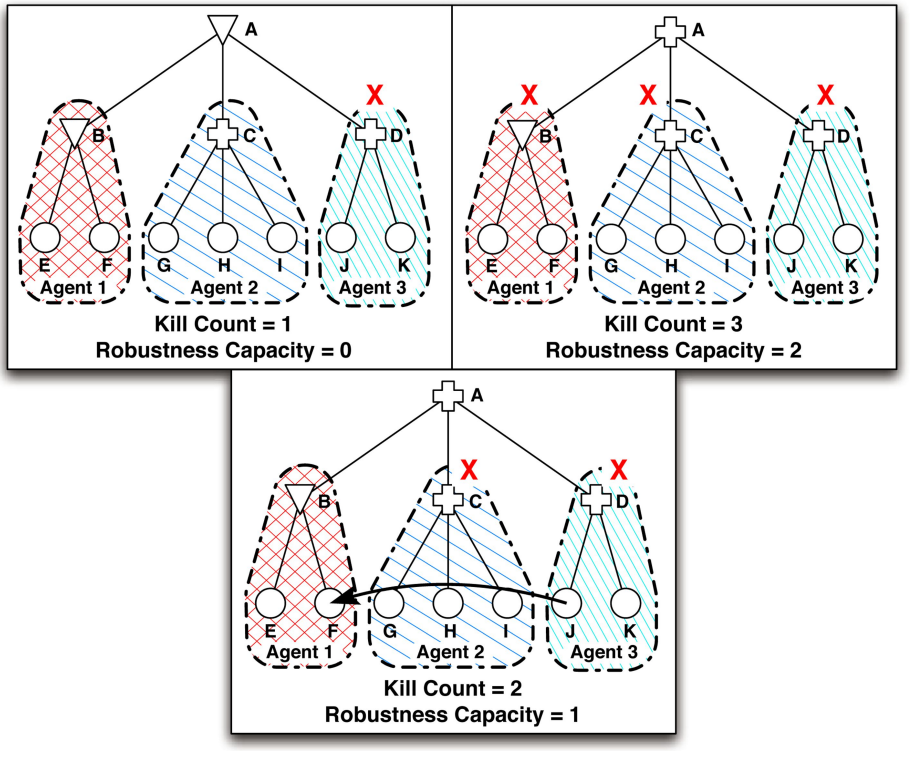


Fig. 3. Computing Robustness Capacity: Figure showing how the global task structure and its breakup amongst the agents affects the robustness capacity. The crosses on the agents show which agents need to be killed in order to kill the organization.

(i.e. ensure that the remaining agents cannot complete tasks without having to restart more agents).

The robustness capacity of an organization is dependent on (a) the underlying global task structure and (b) the way it has been divided amongst the agents. The CAFs of the global task structure, especially the root CAF, determines the number of alternatives available for achieving a task. For example, a one-level deep task structure with a *MAX* CAF and three subtasks would have three alternative ways of achieving the task. If each of these three alternatives was divided amongst three agents, the resultant organization would have a kill count of 3 and a robustness capacity of 2.

Figure 3 shows how the global task structure and its breakup amongst the agents affects the robustness capacity of an organization. The first task structure has a *MIN* CAF as its root, so *either* of Agents 1, 2 or 3 can be killed in order to kill the organization. The kill count is 1 and, hence, the robustness capacity is 0. The second task structure has a *SUM* as its root, so *all* the three agents need to be killed in order to kill the organization. Hence, the kill count is 3 and the robustness capacity is two. The third task structure is similar to second one except that it has an enablement from Method J to F. With the task structure divided amongst the agents as shown, if Agent 3 is killed, there is no way

to complete Method J. This, in turn, means that Method F will never be enabled, i.e. the quality of Method F will always be 0. Since Task B has a *MIN* CAF, the quality of Task B will also be 0 and, as a result, Agent 1 has effectively been poisoned. Hence, only Agents 2 and 3 need to be killed to kill the organization and the kill count is 2.

Augmenting the robustness capacity of an organization is the process of adding agents to the organization so as to increase its kill count. Again, we are assuming that the desired kill count, *K* is an input to the organization. A trivial way to do this would be to replicate each agent *K-1* times. However, this would be inefficient as it does not take into account the existing kill-count of the organization.

The first step towards increasing the robustness capacity of an organization would be to compute the existing kill-count, and then to "add" agents by breaking up the global task structure and spawning agents in a way that increments this kill-count. Unfortunately, the bad news is that computing the kill-count of an organization based on an underlying TÆMS task structure is NP-hard. An informal proof follows:

This proof is based on the reduction of a minimum set covering problem to a TÆMS based organization, where the kill-count of that organization would be the solution to this problem. Assume a ground set *M* consisting of m elements, $\{e_1, e_2, ..., e_m\}$ and n subsets $\{s_1, s_2, ..., s_n\}$. Create a TÆMS task structure, with a *MAX* CAF as the root and the subsets, $\{s_1, s_2, ..., s_n\}$ as its subtask nodes. Finally replace each node s_i with a *MIN* CAF task, the subtasks of which will be the methods, $\{e_{i,1}, e_{i,2}, ..., e_{i,j}\}$, where each method corresponds to an element of s_i. Finally, assign m agents to the organization, where each method corresponding to e_i is assigned to agent a_i. This reduction will provably take polynomial time.

Since, (a) the problem of computing the kill count/robustness capacity of a problem is NP-hard and (b) augmenting the organization by spawning agents at specific places will interfere with other desirable characteristics such as balancing the execution time and maximizing quality, we chose an alternative approach to augmenting the robustness capacity.

In our approach, the initial root node of the global task view is cloned $K-1$ times and each clone is allocated to a separate agent. These agents are responsible for individually forming their own independent organizations and spawning and composing with agents independently.

Frequency of *are-you-alive* messages. Ideally, we want each agent in the monitoring set of an agent A to send an *are-you-alive* request at a different time. To achieve this, we initialize each agent with a random seed. The *next-poll-time* is initialized to the *poll-interval* plus this random seed. Also the next-poll-time is recalculated on receiving *any* message from the monitored agent.

5 Evaluation

To evaluate the two robustness mechanisms, we ran a series of experiments that simulated the operation of the OSD organization when those mechanisms were employed. We tested the performance of the survivalist approach against the citizens approach with the following (per agent/per cycle) probabilities of agent failures: 0.000, 0.002, 0.006

and 0.010. Here a probability of 0.006 means that on every clock cycle, each agent has a 0.6 % chance of failing. Note that despite these seemingly low probabilities of failure, the rate of failure is actually greater than can be expected for any real world application. For example, a probability of failure of 0.006 implies that *every* agent can be expected to fail 15 times during a 2500 cycle run.

We used a randomly generated TÆMS task structure with a maximum depth of 4, branching factor of 3, and NLE count of 10 to seed the experiments. We were careful to use the same task structure, task arrival times, task deadlines and random numbers for each of the (robustness-mechanism, failure probability) pairs. Each experiment was repeated 15 times using a different randomly generated task structure. The experiments were run for 2500 clock cycles. For the survivalist approach we used 3 as the value of K, i.e. the root node is cloned *twice*. We used the following performance criteria to evaluate our two approaches to robustness:

1. The average number of agents used.
2. The number of tasks completed.
3. The average turnaround time. The turnaround time is defined as the difference between the time at which a task is either completed or failed and the time at which the task was generated (the generation time). The average turnaround time is the turnaround time divided by the total number of tasks.
4. The average quality accrued. The average quality is defined as the total quality accrued during the experimental run divided by the sum of the number of tasks completed and the number of tasks failed.
5. The total messages sent by all the agents.
6. The total resource cost of the organization.

The average results for these measured performance criteria are shown in Figure 4.

We also tested the statistical significance of the obtained results using the *Wilcoxon Matched-Pair Signed-Rank* tests with $p < 0.05$. *Matched-Pair* signifies that we are comparing the performance of each robustness approach on precisely the same randomized task set, environmental conditions and failure probabilities within each separate experiment. Some interesting observations are:

- We were pleasantly surprised by the overall performance of these approaches. In particular there was no statistical difference between the number of tasks completed in the absence of agent failure and all the situations in which the citizens approach was employed. Only the survivalist approaches at probabilities of failure of 0.006 and 0.010 performed statistically worse than the no failure case. This result shows that the survivalist approach is a credible distributed alternative to the citizens approach.
- As expected the fewest number of agents were used when the probability of failure was 0. Surprisingly, not only did the survivalist approach use fewer agents than the citizens approach, this result was *statistically significant* for failure probabilities of 0.006 and 0.010. This results is surprising because we expected the survivalist approach to use more agents since the survivalist approach pro-actively replicates agents to increase the robustness capacity of the organization. The reasoning behind this result becomes obvious once we look at the percentage of the total tasks completed by the survivalist organizations. The survivalist organizations complete

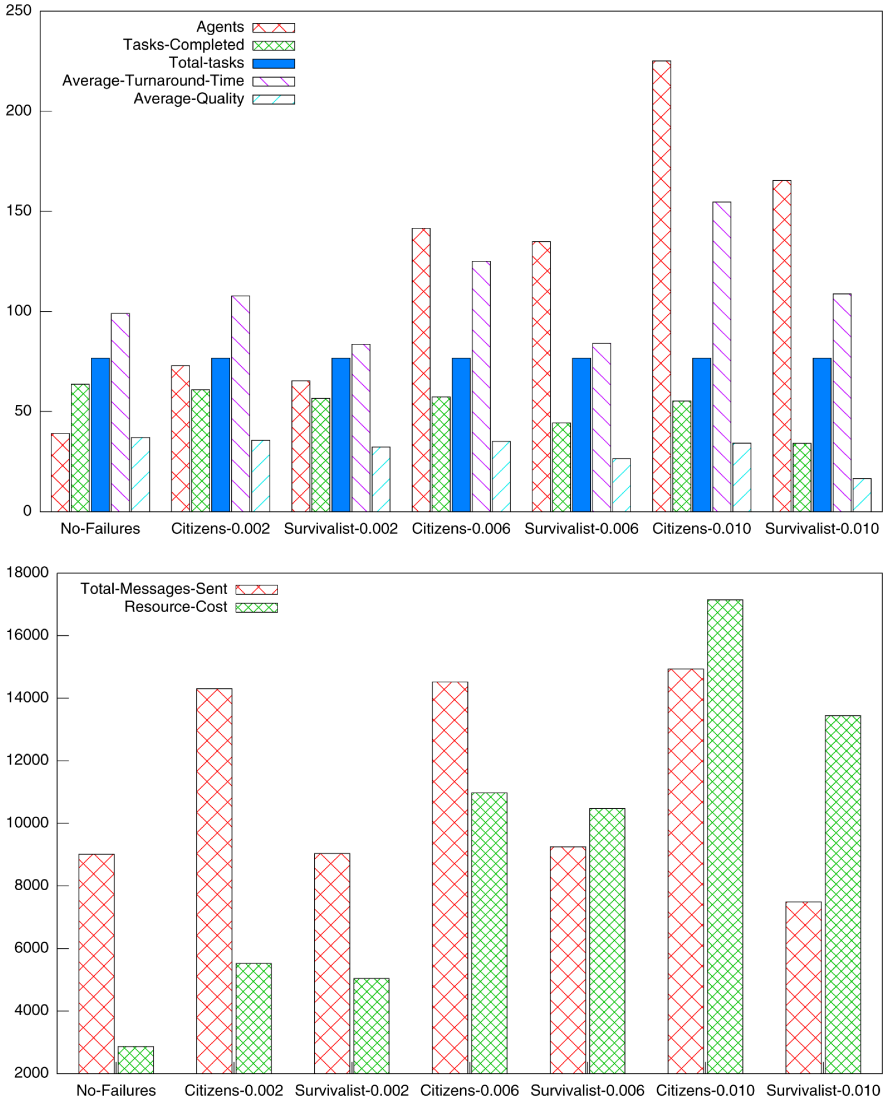


Fig. 4. Graph showing the various measured parameters for the different robustness mechanisms. The numbers below the mechanisms indicate the probability of agent failure per agent per cycle.

fewer tasks for failure probabilities of 0.006 and 0.010. This is because the survivalist organizations are being overwhelmed by the failure rate – the agents are failing much faster than the rate at which the organization can detect the failures and restart the failed agents.

Using a higher value of K, should allow the survivalist approach to perform as well as the citizens approach. As a part of our future work, we would like to automatically determine the best value of K at run-time.

- The citizens approach used a significantly larger number of messages to achieve similar levels of robustness. This is primarily because of the monitor agent has to record all the messages exchanged by the agents.
- Finally, the turnaround time is statistically significantly lower for the survivalist approaches with probabilities of 0.002 and 0.006 than all the other approaches including the no failure approach. This is probably the result of extra agents better balancing the load of the organization.

6 Conclusion

This paper was primarily concerned with the robustness of organizations, generated at run-time through the use of organizational-self design (OSD), in the presence of agent failures. We have incorporated the two commonly used approaches to robustness, that is, the citizens approach and the survivalist approach into our OSD system. The citizens approach is simpler and more effective than the survivalist approach but suffers due to the use of a single centralized and omniscient monitoring agent to achieve its robustness. The survivalist approach, on the other hand, is truly distributed and we have shown it to be a credible alternative to the citizens approach since it uses fewer agents and fewer (communication) resources to achieve similar levels of robustness.

In our future work, we would like to develop a more fine-grained method for augmenting the robustness capacity of an organization in the survivalist approach — one that will use some heuristic to compute or *underestimate* the current robustness capacity of the organization and then augment the robustness capacity by cloning specific agents. We would like to see if such a fine-grained approach will use even fewer agents than the currently presented survivalist approach.

References

1. Dellarocas, C., Klein, M.: An experimental evaluation of domain-independent fault handling services in open multi-agent systems. In: Proceedings of the International Conference on Multi-Agent Systems (ICMAS 2000) (July 2000)
2. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. In: Distributed Artificial Intelligence, pp. 357–366. Morgan Kaufmann Publishers Inc., San Francisco (1988)
3. Ishida, T., Gasser, L., Yokoo, M.: Organization self-design of distributed production systems. IEEE Transactions on Knowledge and Data Engineering 4(2), 123–134 (1992)
4. Kamboj, S., Decker, K.S.: Organizational self-design in semi-dynamic environments. In: AAMAS 2007, pp. 1220–1227 (2007)
5. Anderson, D.P.: Boinc: A system for public-resource computing and storage. In: Fifth IEEE/ACM International Workshop on Grid Computing (GRID 2004), pp. 4–10 (2004)
6. Shirts, M., Pande, V.S.: COMPUTING: Screen Savers of the World Unite! Science 290(5498), 1903–1904 (2000)
7. Wagner, T., Raja, A., Lesser, V.: Modeling uncertainty and its implications to sophisticated control in taems agents. Autonomous Agents and Multi-Agent Systems 13(3), 235–292 (2006)
8. DeLoach, S., Oyenan, W., Matson, E.: A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems 16(1), 13–56 (2008)

9. Sims, M., Goldman, C.V., Lesser, V.: Self-organization through bottom-up coalition formation. In: AAMAS 2003, pp. 867–874. ACM Press, New York (2003)
10. So, Y., Durfee, E.H.: Designing tree-structured organizations for computational agents. Computational and Mathematical Organization Theory 2(3), 219–245 (1996)
11. Frederic, G., Jacqueline, A.: Logical reorganization of DAI systems. LNCS. Springer, Heidelberg (1995)
12. Klein, M., Rodriguez-Aguilar, J.A., Dellarocas, C.: Using domain-independent exception handling services to enable robust open multi-agent systems: The case of agent death. Journal for Autonomous Agents and Multi-Agent Systems 7(1-2), 179–189 (2003)
13. Marin, O., Sens, P., Briot, J., Guessoum, Z.: Towards adaptive fault tolerance for distributed multi-agent systems. In: Proceedings of European Research Seminar on Advances in Distributed Systems (ERSADS 2001) (May 2001)
14. Lesser, V.R., Decker, K., Wagner, T., et al.: Evolution of the GPGP/TÆMS Domain-Independent Coordination Framework. Autonomous Agents and Multi-Agent Systems 9(1-2), 87–143 (2004)
15. Chen, W., Decker, K.S.: The analysis of coordination in an information system application - emergency medical services. In: Bresciani, P., Giorgini, P., Henderson-Sellers, B., Low, G., Winikoff, M. (eds.) AOIS 2004. LNCS, vol. 3508, pp. 36–51. Springer, Heidelberg (2005)
16. Decker, K.S.: Environment centered analysis and design of coordination mechanisms. Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst (May 1995)
17. Decker, K.S., Li, J.: Coordinating Mutually Exclusive resources using GPGP. Autonomous Agents and Multi-Agent Systems 3(2), 133–157 (2000)
18. Zimmerman, T.L., Smith, S., Gallagher, A.T., Barbulescu, L., Rubinstein, Z.: Distributed management of flexible times schedules. In: Sixth AAMAS (May 2007)
19. Shehory, O., Sycara, K., Chalasani, P., Jha, S.: Agent cloning: an approach to agent mobility and resource allocation. IEEE Communications Magazine 36(7), 58–67 (1998)

Instrumenting Multi-agent Organisations with Artifacts to Support Reputation Processes

Jomi Fred Hübner*, Laurent Vercouter, and Olivier Boissier

ENS Mines Saint Etienne / G2I / SMA
158 Cours Fauriel
42023 Saint-Etienne Cedex, France
{hubner,boissier,vercouter}@emse.fr

Abstract. Reputation is often cited as an instrument to enforce norm compliance: agents that do not follow the norms have their reputation decreased. Conceiving reputation as a collective process, i.e. a kind of *shared voices* as proposed by Conte & Paolucci, is not a simple task. In this paper, we propose a first step in this direction by instrumenting multi-agent organisation with an *artifact* that publishes some objective evaluations of the performance of the agents with respect to their behaviour within the organisation. The members of the organisation can then read these evaluations and build up their reputation of others. The artifact serves thus as an instrument that aid in the building of the reputation of the agents. We propose that the evaluation of the agents is not simply based on their obedience to norms, but also considers their pro-activeness and their contribution to the success of collective tasks that are being executed in the organisation. This proposal is detailed and exemplified in the context of the $\mathcal{M}$OISE$^+$ organisational model supported by a set of organisational artifacts as proposed in the ORA4MAS approach.

Keywords: organisation, artifacts, norm enforcement, reputation.

1 Introduction

The concept of multi-agent *organisation* is becoming widely accepted as an instrument for open systems not only to help the coordination of autonomous agents but also to control their autonomy [3,13]. For example, when someone adopts the role of master student in a laboratory, she remains autonomous to perform its research but should follow some rules of the laboratory organisation. These rules vary from 'the access to computers requires an username' to 'a master thesis should be written in two years'. The agent is free to adopt the role, but once adopted the organisation expects her autonomy to be limited.

An important feature of this approach when applied to multi-agent systems (MAS) is the flexibility: the agents are neither completely autonomous to do whatever they want nor completely constrained to pre-defined behaviours. The

* Supported by ANR Project ForTrust (ANR-06-SETI-006).

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 96–110, 2009.

organisation serves as a kind of 'tuning' of the autonomy level. To find out a good degree of allowed autonomy is indeed a challenge, specially in the case where the agents have to organise the system themselves [11].

The success of this organisational approach depends on how the compliance to the rules is ensured inside the system. An approach to deal with this issue is to use the agent's *reputation* as an instrument to enforce the compliance to organisational rules. The general proposal is that the agent's behaviour is constantly evaluated by the organisation with respect to the roles it plays and the result of this evaluation is published to other members (phase *i*). This information helps then the agents to construct the reputation of others inside the organisation (phase *ii*). Hence the reputation influences decision processes (e.g. when agents have to select partners to cooperate with), agents take care of their reputation and behave accordingly (phase *iii*). While phases *ii* and *iii* are concerned with how the agents will use the published information, the first phase can be conceived outside the agents. The main contribution of this work is to describe how this first phase of the process can be instrumented in a multi-agent organisation by using artifacts as proposed in ORA4MAS [15].

The next section presents a general analysis of the norm enforcement in the context of organisations and the main concepts used in this paper. In the sequence (Sec. 3), these concepts are reified on the $\mathcal{M}$OISE$^+$ organisational model on which our proposal is based both at the modelling language level and at the organisation management level. Our proposal (to use artifacts for supporting reputation processes in open organisations) is then detailed in Sec. 4. We finish the paper discussing related works, specially those that consider the relation between organisation and reputation.

2 Norms in Multi-agent Organisations

To illustrate the concepts used in the sequence of this section, we will use the following scenario:

> Alice has recently started her master's course in a French research laboratory in computer science. As a master student she has thus several norms to follow: write a technical report from state of the art in the thesis' subject; write a paper in English, code programs to experiment ideas, be friendly with colleagues, use only computers allocated to the master course, do not break equipments, etc. Alice also plans to continue her studies in a PhD course in the same laboratory. She is thus concerned about her reputation during the masters because it is normally used in the PhD selection process.

In this scenario, we can identify several *norms* that limit the autonomy of Alice and that she accepted when entered in the master course. Roughly a norm is an obligation, permission, or interdiction to perform some action or achieve

some goal. A norm may also have a condition that states when it is active and a deadline to be fulfilled (write a thesis in three years).[1] Despite the disposition of the agents to follow these norms or not, the organisation should have instruments to ensure that they are followed. Before presenting these instruments, the next subsection presents two types of mechanism to implements them.

2.1 Regimentation and Enforcement

In the above scenario, we can distinguish a sort of norms that can be ensured by the organisation itself. For example, the norm 'use only computers allocated to the master course' can be ensured by user's profiles and passwords. In this example, the login procedure to access computers is the *instrument* that implements that norm. However, norms like 'write a paper in English' do not need to (or cannot) be forced by the organisation. The organisation cannot force students to write a paper in the same way it can force them to access only authorised computers. We are thus considering two main mechanisms to implement norm in a MAS:[2]

- *Regimentation* is a mechanism that simply prevents the agents to perform actions that are forbidden by a norm. More precisely, we regiment some actions in order to preserve important features of the system (e.g. the access to the computers).
- *Enforcement* is a mechanism which is applied after the detection of the violation of some norm. While regimentation is a preventive mechanism, enforcement is a reactive one. From the point of view of the agents, they may decide to obey or not the norm according to their local view of the organisation. From a system point of view, the fulfilled/unfulfilled of the norms should be detected, evaluated as a violation or not, and then judged as worth of sanction/reward or not.

These two mechanisms allow us to balance (i) the ensuring of very important properties of the system by means of regimentation and, by means of enforcement, (ii) the agents' autonomy required to keep the possibility to adapt and evolve. The norms of the MAS can be instrumented either as regimentations or enforcement mechanisms depending on which side the designer wants to give more weight. Briefly, regimentation should be used to fully constrain the actions of the agents and enforcement should be used when some violation is allowed (or even desired).

[1] We are aware that the concept of norm is broader and more complex than that used in this paper (e.g. [21] and the Deontic Logic in Computer Science workshop series [7]). For the present paper however this simple and informal definition is enough to discuss the proposal.

[2] This classification is based on the proposal described in [9,6]. However, we present them in a more specific context: regimentation is applied only to the interdiction of organisational actions and enforcement is applied to the other cases.

2.2 Norm Management

In the context of platforms for MAS organisation the regimentation mechanism is often used. Agents run on an infrastructure that ensures that all norms will be respected, as in the case of AMELI [5] where norms are ensured by means of governors, $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [12] by organisational proxies, and ORA4MAS [15] by organisational artifacts. For example, when an agent sends a message in the context of a protocol execution, if the message does not follow the rules stated by the protocol, the message is not indeed sent. The action of the agent is not executed since it cannot violate the norm entitled by the protocol. Organisational infrastructures normally use regimentation as an instrument to implement its norms.

Some organisational models have however norms that cannot be implemented by regimentation. In the $\mathcal{M}$OISE$^+$ model, for example, two roles may be related by an authority link: the agent playing the role ρ_1 has to obey orders from the agent playing the role ρ_2 [13]. It is very difficult to have instruments that regiment this norm in a MAS, specially in open systems where the internal state of the agents is neither visible nor controllable by the organisation. Enforcement mechanisms are thus required to implement the norms in cases like that.

The enforcement mechanism normally considers two main steps: violation detection and sanction application. The detection of violation is certainly a hard task in MAS and several proposals have been presented (e.g. [22]). However, as stressed in [9] detection without sanction is worthless. The problem we identified and that motivated our work is that, as far as we know, no organisational platform consider the sanction issue.

In this paper, we propose to instrument the organisation with an artifact that could help in the first phase of a sanction system based on reputation, as described in the introduction: evaluation of the behaviours of agents within an organisation. The proposed artifact is detailed on a particular organisational model: $\mathcal{M}$OISE$^+$. The next section thus briefly describes this model based on an example and identifies some of its norms. The section also describes how they are managed within the ORA4MAS approach.

3 The $\mathcal{M}$oise$^+$ Organisational Model and Its Artifacts

The $\mathcal{M}$OISE$^+$ model proposes an organisational modelling language that explicitly decomposes the specification of organisation into structural, functional, and deontic dimensions [13]. The structural dimension specifies the *roles*, *groups*, and *links* of the organisation. The definition of roles states that when an agent decides to play some role in a group, it is accepting some behavioural constraints related to this role. The functional dimension specifies how the *global collective goals* should be achieved, i.e. how these goals are decomposed (in global *plans*), grouped in coherent sets (by *missions*) to be distributed to the agents. The decomposition of global goals results in a goal-tree, called *scheme*, where the leaves-goals can by achieved individually by the agents. The deontic dimension

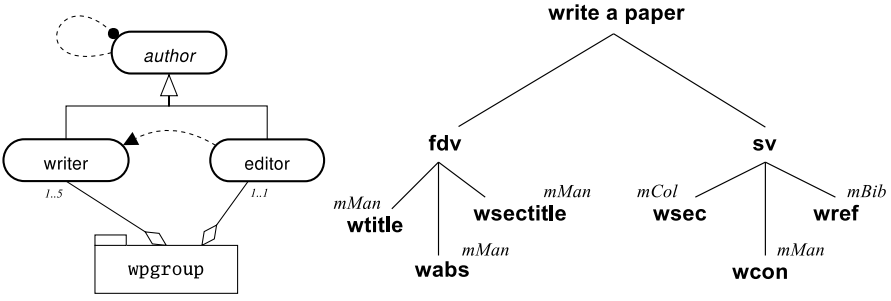
is added in order to bind the structural dimension with the functional one by the specification of the roles' *permissions* and *obligations* for missions.

As an illustrative and simple example of an organisation specified using $\mathcal{M}$OISE$^+$, we consider agents that aim at writing a paper and therefore have an organisational specification to help them to collaborate. The structure of this organisation has only one group (`wpgroup`) with two roles (`editor` and `writer`) that inherit all properties defined for the role `author`. The cardinalities and links of this group are specified, using the $\mathcal{M}$OISE$^+$ notation, in Fig. 1(a): the group `wpgroup` can have from one to five agents playing `writer` and exactly one playing `editor`; the `editor` has authority over `writer` and every agent playing `author` (and by inheritance everyone playing `writer` or `editor`) has the possibility to communicate with every agent playing `author` (communication link from `author` to `author`). In this example, the `editor` and the `author` roles are not compatible. To be compatible, a compatibility relation must be explicitly added in the specification.

To coordinate the achievement of the goal of writing a paper, a scheme is defined in the functional specification of the organisation (Fig. 1(b)). In this scheme, a draft version of the paper has to be initially defined (identified by the goal *fdv* in Fig. 1(b)). This goal is decomposed into three sub-goals: write a title, an abstract, and the section titles. Other agents then 'fill' the paper's sections to get a submission version of the paper (identified by the goal *sv*). The goals of this scheme are distributed in three missions which have specific cardinalities (cf. Fig. 1(c)): $mMan$ for the general management of the process (one and only one agent can commit to it), $mCol$ for the collaboration in writing the paper's content (from one to five agents can commit to it), and $mBib$ for getting the references for the paper (one and only one agent can commit to it). A mission defines all goals an agent commits to when participating in the execution of a scheme, for example, commit to the mission $mMan$ is indeed a commitment to achieve four goals of the scheme. Goals without an assigned mission are satisfied by the achievement of their subgoals. The deontic relation from roles to missions is specified in Fig. 1. For example, any agent playing the role `editor` is permitted to commit to the mission $mMan$.

The specification of an organisation is written in a suitable language, that the agents are supposed to interpret. This language is founded on components represented by predicates and functions. We present here only those components that are used in the sequel of the paper. Considering an organisational specification, $\mathcal{G}$ is the set of all group specifications, $\mathcal{R}$ is the set of all roles, $\mathcal{S}$ is the set of all scheme specifications, $\mathcal{M}$ is the set of all missions, and Φ is the set of all goals.

- $compat(g, \rho, C)$: is a predicate that is true when the role ρ ($\rho \in \mathcal{R}$) is compatible with all roles in the set C ($C \subseteq \mathcal{R}$) when played in the group g ($g \in \mathcal{G}$);
- $mission\_scheme(m, s)$ is a predicate that is true when the mission m ($m \in \mathcal{M}$) belongs to the scheme s ($s \in \mathcal{S}$);
- $goal\_mission(\varphi, m)$: is a predicate that is true when the goal φ ($\varphi \in \Phi$) belongs to the mission m ($m \in \mathcal{M}$);



(a) Structural Specification (b) Functional Specification

mission	cardinality
$mMan$	1..1
$mCol$	1..5
$mBib$	1..1

(c) Missions

role	deontic relation	mission
editor	*permitted*	$mMan$
writer	*obliged*	$mCol$
writer	*obliged*	$mBib$

(d) Deontic Specification

Fig. 1. Graphical representation of the organisational specification for the writing paper example with the $\mathcal{M}$OISE$^+$ OML

- $obl(\rho, m)$: is a predicate that is true when the role ρ has an obligation relation to the mission m;
- $goal\_role(\varphi, \rho)$: is a predicate that is true when the role ρ is obliged to the goal φ, this predicate is defined as follows

$$goal\_role(\varphi, \rho) \leftrightarrow goal\_mission(\varphi, m) \ \wedge \ obl(\rho, m)$$

3.1 ORA4MAS: Managing Organisation with Organisational Artifacts

The $\mathcal{M}$OISE$^+$ model is implemented, on one hand, by an organisational modelling language to program declarative organisation specifications, and, on the other hand, by organisational artifacts, as those proposed in ORA4MAS approach [15], that interpret the specification and manage the organisation. The conception of the artifacts follows the A&A (Agents and Artifacts) model [18]. In this model, the environment is not a merely passive source of agent perceptions and target of agent actions, but a first-class abstraction that can be suitably designed to encapsulate some fundamental functionalities and services, supporting MAS dimensions such as coordination and organisation. In particular A&A introduces a notion of *artifact* as first-class abstraction representing function-oriented dynamic entities and tools that agents can create and use to perform their individual and social activities. Thus, while agents are goal-oriented pro-active entities, artifacts are function-oriented passive entities, designed by MAS designers to encapsulate some kind of functionality, by representing (or wrapping existing) resources or instruments mediating agent activities.

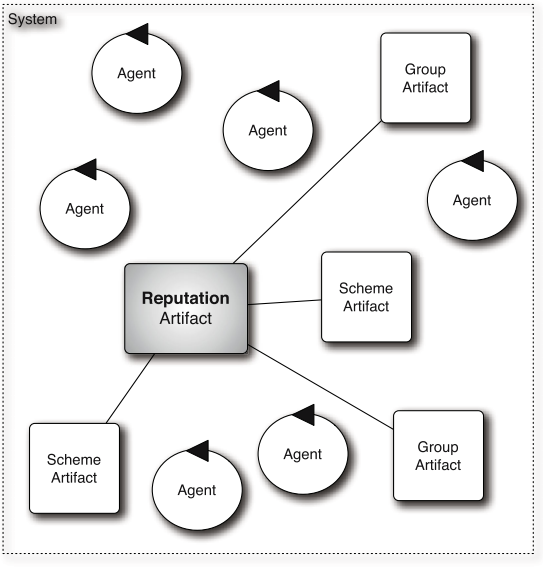


Fig. 2. Agents and Organisational Artifacts

Each artifact is mainly composed of two interfaces: usage and link interfaces. The *usage interface* include (1) a set of operations that agents can trigger to get artifact services and behaviours, and (2) a set of *observable properties* that the agents can inspect (observe) without necessarily executing operations on it. The execution of an operation upon an artifact can result both in changing the artifact's inner (i.e. non-observable) state, and in the generation of a stream of *observable events* that can be perceived by agents that are using or simply observing the artifact. The *link interface* provides operations to another artifact enabling composed functionalities. Agents exploit artifacts functionality (that is, they *use* artifacts) by acting on artifact usage interface which functions as a control panel, and can be aware of artifact observable state by observing observable properties.

As depicted in Fig. 2, agents are situated in an environment with artifacts that they can use for different services. In the particular case of ORA4MAS, we are emphasising the organisational artifacts that offer all organisational services required in an organisational management platform. There are three main types of artifacts in the figure: group, scheme, and artifacts for the reputation processes. The latter will be explained in the next section. *Group artifacts* maintains the state of an instance of group type and offer operations related to this group. For example, when an agent wants to adopt a role in a group, she should go to the corresponding artifact and trigger the `adoptRole` operation. Similarly, a *scheme artifact* offers operations related to the execution of an instance of a scheme, e.g. commitment to missions. As observable properties, the group artifact shows the current players of the group and the scheme shows the players and possible goals. More precisely, from the observable properties of all organisational artifacts, we can define the following sets, predicates, and functions:

- $\mathcal{A}$: the set of all agents inside the organisation;
- $plays(\alpha, \rho, g)$: it is true that the agent α plays the role ρ in the group g (g is an instance of a group in $\mathcal{G}$);
- $committed(\alpha, m, s)$ it is true that the agent α is committed to the mission m in the scheme s (s is an instance of a scheme in $\mathcal{S}$);

- *achieved*(φ, α): it is true that the goal φ is already achieved by the agent α;
- *possible*(φ): it is true that the goal φ is possible. Possible goals are those that are not achieved yet and that all pre-condition goals are satisfied. For example, the goal 'to write the conclusion of the paper' can be achieved only after the goal of writing sections was achieved;
- *succeeded*(s) it is true that the scheme s has finished successfully.

Besides providing operations and information to the agents, ORA4MAS artifacts are also responsible to (i) ensure that all regimented norms are followed by the agents and (ii) detect the violation of norms. However they do not implement the violation policies that conclude to sanctions (these policies are delegated to organisational agents). All violated norms can be both displayed as observable properties of the artifact and sent to the reputation artifact. As well as a clear separation of concerns between agents and artifacts, the A&A approach simplifies the decentralisation of the infrastructure once one artifact is loosely coupled to others.

3.2 $\mathcal{M}$oise$^+$ Norms

Based on the $\mathcal{M}$OISE$^+$ specification and a platform like ORA4MAS that provides runtime information of the current state of the organisation, we can write several norms to constrain the agent's behaviour. However it is not the focus of this paper to present how the overall organisational specification is translated to norms. Two examples are thus presented to illustrate the use of norms in a $\mathcal{M}$OISE$^+$ based organisation. In the following these norms are represented as a pair where the first argument is the condition part stating when the norm is active and the second argument is the action part stating an obligation, permission, or interdiction.

Example 1: roles are incompatible unless explicitly stated the contrary in the specification. Thus, if it is stated that two roles ρ_1 and ρ_2 are compatible inside a group g ($compat(g, \rho_1, \{\rho_2\})$), it implies that an agent that plays ρ_1 in the group g cannot perform the operation `adoptRole(`ρ_i, g`)` for any $i \neq 2$. This constraint on role adoption is formalised by the following norm:

$$\begin{array}{l} (plays(\alpha, \rho, gb) \ \wedge \ compat(g, \rho, C), \\ \forall_{\rho_i \in \mathcal{R} \setminus C} \ forbiden(\alpha, \texttt{adoptRole}(\rho_i, g))) \end{array} \quad (1)$$

The condition of the norm (the first line) is a conjunction of predicates. Its evaluation is given by the particular circumstance of the group (that defines whether $plays(\alpha, \rho, gb)$ holds or not) and the structural specification being used (that defines whether $compat(g, \rho, C)$ holds or not). The action part of the norm (the last line) states that it is forbidden for agent α to execute the action `adoptRole` on any role that does not belong to the set of compatible roles C. Based on this norm, as soon as an agent adopts a role (activating the norm), the adoption of other roles that are not explicitly stated compatible are forbidden for it.

Once this two norm is of the type 'action interdiction', they can be easily implemented by regimentation: whenever the `adoptRole` operation is requested by the agent α, if the condition of the norm holds, the execution of the corresponding operation is denied.

Example 2: once an agent α is committed to a mission m, it is obliged to fulfil the possible goals of the mission. The norm below specifies that rule.

$$(committed(\alpha, m, s) \wedge goal\_mission(\varphi, m) \wedge possible(\varphi, s), \\ obliged(\alpha, \varphi)) \tag{2}$$

While the first norm can be easily implemented in the organisational artifacts (`adopt_role`(ρ, g) is an organisation action under the control of the artifact), the implementation of this latter example is not so easy: how can we detect that some agent is not pursuing a goal without accessing its internal state; how can we enforce agents to follow their organisational obligations. The next section deal with these problems.

4 Instrumenting Reputation Processes with Artifacts

The reputation is widely cited as an instrument to enforce norms [8,9,16,22]. However few proposals are detailed in the context of an organisational infrastructure that aims to enforce its norms. Inspired by the concept of *reputation artifact* proposed in [2, p. 101], this section details such artifact in the context of the ORA4MAS approach. It provides first class constructs which can be easily used to enrich the support of reputation processes.

4.1 Agent's Reputation

The new artifact that we propose to add in the system serves as an indirect sanction instrument for norms enforcement. While direct sanctions are applied when the violation is detected, indirect sanctions have long term results, as is the case of reputation.

This very artifact is linked to all organisational artifacts of the ORA4MAS and can be observed by all agents inside the organisation. Other artifacts notify it about the current state of the organisation and then this information is used to compute an *evaluation* for each agent inside the organisation. This evaluation is published as an observable property of the artifact. It is important to notice that the evaluation is not the reputation of the agent, as remarked in [2], reputation is a *shared voice* circulating in a group of agents. This artifact is indeed an instrument to influence the reputation of the agent.

Several criteria may be used to evaluate an agent inside an organisation. Herein we choose to evaluate an agent in the context of the roles and missions she is engaged. Three criteria are used: obedience, pro-activeness, and result.

The *obedience* of an agent is computed by the number of obliged goals an agent achieves. The goals an agent is obliged to achieve are defined by norms

(as that presented in the Example 2). All obliged goals that were not yet achieved are considered as a violation.[3] The general mission obedience function ($o : \mathcal{A} \rightarrow [0,1]$) and the obedience in the context of a particular mission ($o_m : \mathcal{A} \times \mathcal{M} \rightarrow [0,1]$) and role ($o_r : \mathcal{A} \times \mathcal{R} \rightarrow [0,1]$) are calculated as follows (in the equations # is a function that returns the size of a set):

$$o(\alpha) = \frac{\#\{\varphi \mid obliged(\alpha,\varphi) \ \wedge \ achieved(\alpha,\varphi)\}}{\#\{\varphi \mid obliged(\alpha,\varphi)\}}$$

$$o_m(\alpha,m) = \frac{\#\{\varphi \mid obliged(\alpha,\varphi) \ \wedge \ goal\_mission(\varphi,m) \ \wedge \ achieved(\alpha,\varphi)\}}{\#\{\varphi \mid obliged(\alpha,\varphi) \ \wedge \ goal\_mission(\varphi,m)\}}$$

$$o_r(\alpha,\rho) = \frac{\#\{\varphi \mid obliged(\alpha,\varphi) \ \wedge \ goal\_role(\varphi,\rho) \ \wedge \ achieved(\alpha,\varphi)\}}{\#\{\varphi \mid obliged(\alpha,\varphi) \ \wedge \ goal\_role(\varphi,\rho)\}}$$

$o(\alpha) = 1$ means that the agent α achieved all its obligation and $o(\alpha) = 0$ means she achieved none. $o_m(\alpha,m) = 1$ means that the agent achieved all goals when committed to the mission m, and $o_r(\alpha,\rho) = 1$ means that the agent achieved all goals when playing the role ρ.

The *pro-activeness* of an agent is computed by the number of goals an agent achieves such that she is not obliged to fulfil that goal in a scheme. The general pro-activeness function ($p : \mathcal{A} \rightarrow [0,1]$) and the pro-activeness in the context of a particular mission ($p_m : \mathcal{A} \times \mathcal{M} \rightarrow [0,1]$) and role ($p_r : \mathcal{A} \times \mathcal{R} \rightarrow [0,1]$) are calculated as follows:

$$p(\alpha) = \frac{\#\{\varphi \mid achieved(\alpha,\varphi) \ \wedge \ \neg obliged(\alpha,\varphi)\}}{\#\Phi \ \#\mathcal{S}}$$

$$p_m(\alpha,m) = \frac{\#\{\varphi \mid achieved(\alpha,\varphi) \ \wedge \ \neg obliged(\alpha,\varphi) \ \wedge \ goal\_mission(\varphi,m)\}}{\#\{\varphi \mid committed(\alpha,m,\_) \ \wedge \ goal\_mission(\varphi,m)\}}$$

$$p_r(\alpha,\rho) = \frac{\#\{\varphi \mid achieved(\alpha,\varphi) \ \wedge \ \neg obliged(\alpha,\varphi) \ \wedge \ goal\_role(\varphi,r)\}}{\#\{\varphi \mid committed(\alpha,m,\_) \ \wedge \ goal\_mission(\varphi,m) \ \wedge \ goal\_role(\varphi,r)\}}$$

$p(\alpha) = 1$ means that the agent achieved all goals she is not obliged to (a highly pro-active behaviour) and $p(\alpha) = 0$ means the contrary.

The *results* of an agent is computed by the number of successful execution of scheme where she participates. It does not depend on the achievement of the goals in the scheme. It means the agent somehow share the success of the scheme execution and likely has helped for the success. The general results function ($r : \mathcal{A} \rightarrow [0,1]$) and the results in the context of a particular mission ($r_m : \mathcal{A} \times \mathcal{M} \rightarrow [0,1]$) and role ($r_r : \mathcal{A} \times \mathcal{R} \rightarrow [0,1]$) are calculated as follows:

[3] We still do not consider the temporal dimension of the obligations. For instance, once an obliged goal is possible for an agent, it is violating the corresponding norm until the achievement of the goal because there is not timeout assigned to the obligation.

$$r(\alpha) = \frac{\#\{s \mid committed(\alpha, \_, s) \ \wedge \ succeeded(s)\}}{\#\{s \mid committed(\alpha, \_, s)\}}$$

$$r_m(\alpha, m) = \frac{\#\{s \mid committed(\alpha, m, s) \ \wedge \ succeeded(s)\}}{\#\{s \mid committed(\alpha, m, s)\}}$$

$$r_r(\alpha, \rho) = \frac{\#\{s \mid committed(\alpha, m, s) \ \wedge \ succeeded(s) \ \wedge \ obl(\rho, m)\}}{\#\{s \mid committed(\alpha, m, s) \ \wedge \ obl(\rho, m)\}}$$

$r(\alpha) = 1$ means that all schemes the agent participated have finished successfully and $r(\alpha) = 0$ means the contrary.

Unlike the previous two criteria, the results value of an agent cannot be increased by the agent itself. This evaluation depends on the performance of all agents committed to the same scheme, creating thus a dependence among them. The selection of good partners is therefore important and the reputation artifact could be used for that purpose.

The aforementioned criteria are combined into a single overall evaluation of an agent ($e : \mathcal{A} \rightarrow [0, 1]$) by the following weighted mean:

$$e(\alpha) = \frac{\gamma\, o(\alpha) + \delta\, p(\alpha) + \epsilon\, r(\alpha)}{\gamma \ + \ \delta \ + \ \epsilon}$$

$$e_m(\alpha, m) = \frac{\gamma\, o(\alpha, m) + \delta\, p(\alpha, m) + \epsilon\, r(\alpha, m)}{\gamma \ + \ \delta \ + \ \epsilon}$$

$$e_r(\alpha, \rho) = \frac{\gamma\, o(\alpha, \rho) + \delta\, p(\alpha, \rho) + \epsilon\, r(\alpha, \rho)}{\gamma \ + \ \delta \ + \ \epsilon}$$

The factors γ, δ, and ϵ are used to define the importance of the obedience, pro-activeness, and results values respectively.

All these objective values provided by the reputation artifact can then be used by agents to compute the reputation of others. It is possible that in one organisation where violation is the rule, if you are a strong violator of norms, your reputation is perhaps greater that in an organisation where violation is not at all the rule.

4.2 Example

This subsection illustrates the evaluations performed by the reputation artifact based on a small history of the organisation created to write papers and presented in the second section. Three instances of the scheme were executed as shown in Table 1, the first and third executions have finished with a paper written, but the second has failed. In the first scheme Bob has chosen Alice as a partner and in the second scheme the partner was Marc. Even though all goals were achieved in the second scheme, the overall scheme failed. One possible reason is the competence of Marc to achieve his goals. In the third scheme Bob decided to work with both Alice and Marc. The scheme finished successfully. Note however

Table 1. Example of an history of the organisation

Scheme	Agent	Role	Mission	Achieved Goals	Unachieved goals
s_1	Bob	editor	mMan	wtitle, wabs, wsectitle, wcon	
	Alice	writer	mCol	wsec	
	Alice	writer	mBib	wref	
s_2	Bob	editor	mMan	wtitle, wabs, wsectitle, wcon	
	Marc	writer	mCol	wsec	
	Marc	writer	mBib	wref	
s_3	Bob	editor	mMan	wtitle, wabs, wsectitle, wcon	
	Alice	writer	mCol	wsec, wref	
	Marc	writer	mCol	wsec	
	Marc	writer	mBib		wref

Table 2. Example of observable properties of the reputation artifact

Agent	o_{mMan}	o_{mCol}	o_{mBib}	o_{editor}	o_{writer}	o	p	r	e
Bob	12/12	–	–	12/12	–	12/12	0/18	2/3	0.29
Alice	–	2/2	1/1	–	3/3	3/3	1/18	2/2	0.41
Marc	–	2/2	1/2	–	3/4	3/4	0/18	2/3	0.26

that Marc did not achieve the goal of compiling the references. This task was done by Alice, even though wref was not her goal.

In the Table 2 the evaluation of the three agents are shown. Only the obligation criteria is presented in all contexts (missions and roles), for the others the general evaluation is included in the table. The values used for γ, δ, and ϵ are respectively 1, 5, and 2. With these parameters, pro-activeness is the more important criteria resulting in Alice as having the best evaluation since she was the only one that performed not obliged goals.

5 Related Works

Some works that consider both the organisation and the reputation are concerned to the problem of how an agent can use the position of another agent in a organisation as an evaluation criteria. This approach is well illustrated in the example cited by [4] where a police uniform gives some reputation to an agent wearing it because of the organisation represented by the uniform. The REGRET [19] and FIRE [14] reputation models also take this direction and use the organisation as yet another source of information (as direct interaction and witness) to form the reputation of a target agent. The organisation gives a kind of 'label' (as an uniform or a role) to the agents. Summing up, they have an agent centred approach and thus collective issues like norm enforcement and sanctions are not considered.

On one hand, our proposal is complementary to the approach used in the works cited above given an organisation centred view of the problem. Although we do not consider how the agents build the reputation of others, we provide an objective and detailed source of information to the agents' reputation model. The information published in the reputation artifact has two important features: (*i*) it is not a simple label assigned to agents ('Bob plays editor') but an evaluation of the performance of the agent in an organisational context (role or mission); and (*ii*) it does not depend on a subjective evaluation, but is rather precisely computed. On the other hand, we differ from the agent centred approach placing the reputation artifact *inside* the organisation. It is supposed to be used by agents of the organisation to chose partners and to improve the overall organisational performance, working as norm enforcement instrument.

Another important work in the domain is presented in [10]. They also take an agent centred approach and propose to consider the place of an agent in the organisation in different contexts. The three levels of evaluation described in our evaluation mechanism (general, role, mission) are inspired by their work.

In a recent work, Da Silva et al [20] proposed an approach that considers both an agent and an organisation centred approach. Agents evaluate others regarding the compliance of their behaviour vis-à-vis the norms. The evaluation and the reasons for such evaluation are then sent to the organisation. One advantage of their proposal is that the agents' evaluations are distributed, since they are performed by agents. This feature requires however that the system is also concerned of the reputation of the agent as 'evaluators'. As in our approach the evaluation is performed by the infrastructure, we can assume the correctness and objectiveness of the information. Another difference is that our evaluation is not based only on norm conformity, the pro-activeness of the agents is also taken into account.

Our approach also shares one property with traditional reputation systems as eBay: the centralisation and publication of the information. Although the evaluations of our proposal are published in one artifact, they are computed by several distributed artifacts (scheme and group artifacts). Another difference is that the evaluation is not performed by users but based on precise metrics with a clear meaning.

Although several authors comment that reputation can be increased or decreased as a kind of sanction, they do not tackle the problem of how to increase/decrease reputation. It is a problem specially when considering the definition of reputation as proposed by [2] – reputation is something outside the agents, but known by them. In this case, to change the reputation is neither to simply change a value in a database nor to answer this value when requested (serving as a witness). The public character of the value is important, and it is achieved by our proposal of reputation artifact.

6 Conclusion and Perspectives

This paper presented work in progress that includes reputation as an instrument to enforce norms inside organisations. Its contribution is twofold: (*i*) a detailed

agent evaluation process that considers the agents obedience, pro-activeness and results in three levels (general, role, mission); and (*ii*) the use of artifacts as instruments for an indirect sanction system. The inclusion of pro-activeness leads us to a system that is not based only on obedience, as pointed out for example by [1], sometimes the agents should break the rules. The inclusion of results forces the agents to choose good partners in the execution of collective tasks. To choose good partners, the reputation artifact can be used, improving thus the importance and effect of this artifact. Although we have presented the concept of reputation artifact in the case of ORA4MAS and $\mathcal{M}$OISE$^+$, its application on other infrastructures is straightforward.

As future work, we intend to study "the agents' side" (phases *ii* and *iii* cited in the introduction): how the information provided by the reputation artifact can be concretely used by the reasoning mechanisms of the agents and how the reputation of the agents are formed. We also plan to implement our proposal in an agent programming language where artifacts are well integrated, as those proposed in [17], and perform an evaluation in a real scenario.

References

1. Castelfranchi, C.: Formalizing the informal? dynamic social order, bottom-up social control, and spontaneous normative relations. Journal of Applied Logic 1(1-2), 47–92 (2003)
2. Conte, R., Paolucci, M.: Reputation in Artificial Societies: Social Beliefs for Social Order. Kluwer, Dordrecht (2002)
3. Dignum, V., Dignum, F.P.M.: Modelling agent societies: Co-ordination frameworks and institutions. In: Brazdil, P.B., Jorge, A.M. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 191–204. Springer, Heidelberg (2001)
4. Esfandiari, B., Chandrasekharan, S.: On how agents make friends: Mechanisms for trus acquisition. In: Proceedings of 4th workshop on deception, fraud and trus in agent societies (2001)
5. Esteva, M., Rodríguez-Aguilar, J.A., Rosell, B., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Jennings, N.R., Sierra, C., Sonenberg, L., Tambe, M. (eds.) Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), pp. 236–243. ACM, New York (2004)
6. Fornara, N., Colombetti, M.: Specifying and enforcing norms in artificial institutions. In: Omicini, A., Dunin-Keplicz, B., Padget, J. (eds.) Proceedings of the 4th European Workshop on Multi-Agent Systems, EUMAS 2006 (2006)
7. Goble, L., Meyer, J.-J.C. (eds.): DEON 2006. LNCS, vol. 4048. Springer, Heidelberg (2006)
8. Grizard, A., Vercouter, L., Stratulat, T., Muller, G.: A peer-to-peer normative system to achieve social order. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 274–289. Springer, Heidelberg (2007)
9. Grossi, D., Aldewered, H., Dignum, F.: Ubi Lex, Ibi Poena: Designing norm enforcement in e-institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 101–114. Springer, Heidelberg (2007)

10. Hermoso, R., Billhardt, H., Ossowski, S.: Integrating trust in virtual organisations. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 19–31. Springer, Heidelberg (2007)
11. Hübner, J.F., Sichman, J.S., Boissier, O.: Using the MOISE+ for a cooperative framework of MAS reorganisation. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 506–515. Springer, Heidelberg (2004)
12. Hübner, J.F., Sichman, J.S., Boissier, O.: S-MOISE+: A middleware for developing organised multi-agent systems. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM 2005 and OOOP 2005. LNCS, vol. 3913, pp. 64–78. Springer, Heidelberg (2006)
13. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1(3/4), 370–395 (2007)
14. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: FIRE: An integrated trust and reputation model for open multi-agent systems. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI) (2004)
15. Kitio, R., Boissier, O., Hübner, J.F., Ricci, A.: Organisational artifacts and agents for open multi-agent organisations: "Giving the power back to the agents". In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS, vol. 4870, pp. 171–186. Springer, Heidelberg (2008)
16. Muller, G., Vercouter, L.: Decentralized monitoring of agent communication with a reputation model. In: Falcone, R., Barber, S., Sabater-Mir, J., Singh, M.P. (eds.) Trusting Agents for Trusting Electronic Societies. LNCS, vol. 3577, pp. 144–161. Springer, Heidelberg (2005)
17. Ricci, A., Piunti, M., Acay, L.D., Bordini, R.H., Hübner, J.F., Dastani, M.: Integrating heterogeneous agent programming platforms within artifact-based environments. In: Padgham, L., Parkes, D.C., Müller, J., Parsons, S. (eds.) 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, pp. 225–232. IFAAMAS (2008)
18. Ricci, A., Viroli, M., Omicini, A.: The A&A programming model and technology for developing agent environments in MAS. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS, vol. 4908, pp. 89–106. Springer, Heidelberg (2008)
19. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: First International Conference on Autonomous Agents and Multiagent systems (AAMAS 2002), pp. 475–482 (2002)
20. de Silva, V.T., Hermoso, R., Centeno, R.: A hybrid reputation model based on the use of organization. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 111–125. Springer, Heidelberg (2008)
21. Tuomela, R., Bonnevier-Tuomela, M.: Norms and agreement. European Journal of Law, Philosophy and Computer Science 5, 41–46 (1995)
22. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Norms in multiagent systems: some implementation guidelines. In: Proceedings of the Second European Workshop on Multi-Agent Systems, EUMAS 2004 (2004), http://people.cs.uu.nl/dignum/papers/eumas04.PDF

A Hybrid Reputation Model Based on the Use of Organizations*

Viviane Torres da Silva[1], Ramón Hermoso[2], and Roberto Centeno[2]

[1] Department of Computer Systems and Computation,
University Complutense
viviane@fdi.ucm.es
[2] Artificial Intelligence Group - DATCCCIA,
University Rey Juan Carlos, Madrid, Spain
{ramon.hermoso,roberto.centeno}@urjc.es

Abstract. In this paper we present a hybrid reputation model focused on organizational structures that attempts to solve problems associated with both centralized and decentralized reputation models. Agents in our approach are able not only to evaluate the behavior of others and store reputations values but also to send such information to a centralized mechanism and ask for reputations to this one and to other agents. The main objective of our approach is to allow agents to reason about the reputation values that they receive. Therefore, together with the reputation values, agents store and send information about norms violated and fulfilled and about the facts that contributed to such behavior. Furthermore, this model provides two different types of reputations, as *service provider* that is related to the behavior of an agent while providing a service to other agents and as *reputation source* that is related to the behavior of an agent while providing reputation of others.

1 Introduction

Several reputation models have been proposed with the aim to make available the reputations of agents interacting in multi-agent systems (MAS). The centralized approaches [2,3] provide mechanisms that aggregate the feedback about the behavior of the agents and make available their reputations. Agents executing in those systems are able to *a)* evaluate the behavior of others with whom they have interacted and *b)* provide testimonies to the reputation model about such behavior. This kind of models presents some problems, such as: *i) scalability*, since it could be necessary to store too much information, and *ii)* most of those models does not allow the agents to know the reputation providers, since usually this type of social information is not stored.

* The present work has been partially funded by the Spanish Ministry of Education and Science under projects TIC2003-08763-C02, TIN2006-14630-C03-02 (FPI grants program) and TIN2006-15660-C02-01; by the Juan de la Cierva program; by Comunidad de Madrid under project S-0505/TIC- 407; by the Spanish project "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010) and by Comunidad de Madrid and University Rey Juan Carlos under project URJC-CM-2006-CET-0300.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 111–125, 2009.

On the contrary, the decentralized reputation models such as [7,9,15] emerged in order to solve the problems of centralized models. They usually have no central authority and allow agents to assess reputation values by asking other agents about their past experiences, so solving *i*. Furthermore, agents can reason about those that behave as *reputation source*, i.e., those that are requested to provide reputation about third parties (solving *ii*). However those mechanisms still present some problems, such as: *iii)* how agents find out "good" *reputation sources* - what can become a problem as difficult to solve as finding a "good" counterpart to interact with. And *iv)* how to give incentives (and what kind of incentive) to agents in order to share their opinions with others.

As we have pointed out, there exist some pros and cons from using both centralized and decentralized reputation models. Nevertheless, they also present a problem shared by both of them: the interpretation of reputation values is never provided when agents exchange reputation values (decentralized models) or even when they ask for them to the system (centralized models). Only the reputation values themselves are presented and no information about fulfilled and violated norms or about the facts that have contributed to such fulfillments or violations are shared. Since two different agents can evaluate the same situation in different ways, it is really a hard task to interpret the reputation values and distinguish trustworthy and untrustworthy agents without additional information about their behavior.

In this paper we propose a hybrid reputation system that uses a centralized and a decentralized mechanism by taking the benefits provided by them while trying to solve some of their drawbacks. The main characteristics of our approach are: *a)* agents are able to evaluate the behavior of other agents, store such information, and provide to the organizations (they are not forced to do this) the reputations evaluated and the reasons for such evaluations (violated and fulfilled norms and the facts that have violated or fulfilled the norms); *b)* organizations implement centralized mechanisms and, therefore, are able to store and to provide the reputation values and related information, and also the identification of the agents that have provided the information; *c)* together with the reputation values, agents store and send information about norms violated and fulfilled and about the facts that contributed to such behavior and in addition *d)* the model allows considering two different dimensions for reputation: agents' reputation as *service providers* and agents' reputation as *reputation sources*. While the reputation as *service provider* represents the degree of satisfaction an agent has obtained after performing an interaction, the reputation as *reputation source* is related to the degree of satisfaction an agent obtains after requesting reputation about other agents to a third party, it evaluates the behavior of an agent while providing information about the reputation of others. Our hybrid model tries to solve the underlined problems mentioned above by giving more semantic meaning to traditional reputation techniques.

Our approach is supported by the scope of *organizations*, where agents enact some *roles* in different *interactions* in order to achieve some *goals*. Some benefits can be obtained from using this approach (i.e. in [6]), since organizational

structures provide a semantically richer information when dealing with reputation values and their reasons.

The paper is organized as follows. In Section 2 we present an overview of the proposed hybrid reputation model. Section 3 details the decentralized mechanism and Section 4 presents the centralized one. Section 5 presents some discussions about related work and Section 6 summarizes and points out some future work.

2 Hybrid Model Overview

Organizational approaches are more and more used in order to build Multi-Agent Systems (MAS) since they allow facing complex problems using simple abstractions [8]. Those abstractions can be concepts that structure relationships among organization members, such as *roles* that agents can play and *interactions* that agents can use to communicate to each other, and also constraints, such as *norms* that establish undesirable agents' behavior. In this paper an organization is specified by the following definition:

Definition 1. *Let $\mathcal{O} \equiv (\mathcal{R}, \mathcal{I}, \mathcal{A}, \mathcal{ON})$ be an organization formed by the following essential elements:*

1. *A set $\mathcal{R}$ of* roles *that are involved in these interactions and can be played by agents in $\mathcal{O}$.*
2. *A set $\mathcal{I}$ of* interactions *available for agents within $\mathcal{O}$.*
3. *A set $\mathcal{A}$ of* agents *(or organization members) that play the roles in $\mathcal{O}$.*
4. *A set $\mathcal{ON}$ of* organizational norms *that regulate the behavior of the agents playing roles in $\mathcal{O}$.*

A typical organization establishes a set $\mathcal{R}$ of *roles* as positions that agents have to put themselves in order to achieve some specific goals while interacting with other agents. Therefore, organizations have to be capable of providing a set $\mathcal{I}$ of *interactions* that can be used by agents to interact with each other. In addition, organizations can define a set $\mathcal{ON}$ of *organization norms* that regulate agents behavior by establishing how agents are expected to fulfill their roles in the organizations in terms of *rights* and *duties*.

Although our model assumes that norms can sometimes be violated. Those norms describe actions that agents are *prohibited*, *permitted* or *obligated* [13] to do and the sanctions to be applied in the case of violations and rewards to be provided in the case of fulfillment [14].

Reputation mechanisms are well-known techniques to fight against unexpected behavior (i.e. norm violations) since they provide agents with relevant information about the trustworthiness of others. The aim of this work is twofold: *i)* to present a hybrid reputation model that, based on the organizational structures (i.e. roles, norms, ...), tries to improve the performance of the organization; and *ii)* to improve the accuracy of reputation mechanisms by using these structures, taking into account the semantics they provide.

2.1 Model Architecture

In order to endow agents with a hybrid behavior concerning reputation mechanisms within the organization, we propose the architecture described in Figure 1. There we can observe:

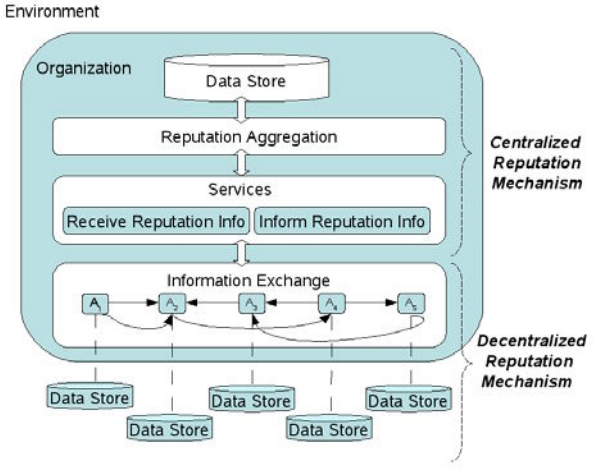


Fig. 1. Model architecture

- A *Centralized Reputation Mechanism* that contains the following entities:
  1. A service that allows agents to send information about others' reputation.
  2. A service that allows the organization to inform agents about others' behavior.
  3. A module that deals with the aggregation of information provided by agents.
  4. A central repository where the information is stored.
- A *Decentralized Reputation Mechanism* composed of the data agents gather by interacting with others.

2.2 Definition of Model Elements

Previous paragraphs pointed out our intention to present a reputation model based on the advantages provided by organizational structures. In this section we stress the relationship between the essential model elements defined in organizations and the agents' reputations.

Reputation of agents is usually associated with the behavior of an agent while playing a role in a given interaction. Therefore, we have defined the concept of *situation* in order to relate agents, interactions and roles, three essential elements of organizations. A similar approach is described in [5,6].

Definition 2. *Let* $\mathcal{S} \equiv \langle \mathcal{A}_i, \mathcal{R}_j, \mathcal{I}_k, \mathcal{O}_o, t \rangle$ *be a* situation *where* $\mathcal{A}_i \in \mathcal{A}$ *is an agent member of the organization* $\mathcal{O}_o \in \mathcal{O}$, $\mathcal{R}_j \in \mathcal{R}$ *represents a role enacted by* $\mathcal{A}_i$ *and* $\mathcal{I}_k \in \mathcal{I}$ *represents an interaction performed by* $\mathcal{A}_i$ *playing the role* $\mathcal{R}_j$, *at instant* t.

A situation can be related to the violation or to the fulfillment of a norm. Both violations and fulfillments are associated with facts that were executed or with facts that should have been executed. In order to state such type of situations we have defined the concept of *(il)legal situation* $(I)LS$.

Definition 3. *Let* $(I)LS \equiv \langle \mathcal{S}_l, \mathcal{N}_m, \mathcal{F}_n \rangle$ *be a* illegal situation ILS *if the situation* $\mathcal{S}_l$ - $\langle A_i, R_j, I_k, O_o, t \rangle$ - *entails a violation of a norm* $\mathcal{N}_m$ *broken by* $\mathcal{A}_i$ *due to some facts represented by* $\mathcal{F}_n$ *or a legal situation* LS *if the situation* $\mathcal{S}_l$ *entails a fulfillment of a norm* $\mathcal{N}_m$ *by* $\mathcal{A}_i$ *due to the facts represented by* $\mathcal{F}_n$.

Most of the reputation systems use quantitative values (opinions) to indicate the reputation of agents. However, such information is not sufficient to understand the behavior of the agents since such values are subjective, i.e. the same norm violation or fulfillment can be differently evaluated by two different agents. The subjective opinion of each agent about the same third party behavior could entail the problem of interpreting the meaning of the agent reputation. In order to tackle this, we propose a reputation model that not only takes into account a numerical value as the opinion an agent provides about a third party behavior, but also the set of norms that the latter has violated or fulfilled and the facts associated with them as a justification of the former's evaluation. This could be viewed as a single-step argumentation about how an agent evaluates the opinion about others.

Definition 4. *Let* $\mathcal{RI} \equiv \langle \mathcal{P}_q, (I)LS_r, Rep \rangle$ *be a* reputation information *provided by the agent provider* $\mathcal{P}_q \in \mathcal{A}$ *about an (il)legal situation* $(I)LS_r \in (I)LS$ *associated with the reputation value* Rep.

The reputation Rep of an agent can vary from $[-1, +1]$. Norm violations are described by illegal situations and are represented by negative reputation values $[-1, 0)$. Norm fulfillments are described by legal situations and are represented by positive reputation values $[0, +1]$.

In order to illustrate the need for stating the norm violated and the facts associated with such violation while informing the reputation of an agent, consider the following example. *Alice* is looking for a seller to purchase a new guitar. In order to choose the most trustworthy one, she decides to ask other agents for opinions about sellers that have sold guitars (or musical instruments) to them. Note that she is searching for opinions about similar situations such as $S_i = \langle Bob, Seller, SellInstruments, \_ \rangle$, where *Bob* is one of the available *sellers* with whom *Alice* is interested to interact using a *SellInstruments* interaction. In the case of seller *Bob* three agents have sent the following reputation values $\langle -0.4, -0.1, -0.5 \rangle$. By analyzing such reputations she can only conclude that *Bob* has violated norms while interacting with those agents but she cannot understand why there exist different reputation values. She has received three different

reputation values about the same agent in similar situations and she does not know what has happened when those agents have interacted with *Bob*.

If *Alice* had received not only the reputation values but also the information about the norms violated and the facts that have violated the norms, *Alice* would be able to understand the different agents' opinions. She would be able to understand that similar violations can be evaluated in different ways and diverse punishments can, thus, be applied by different reputation sources.

2.3 Types of Reputation

In the previous section we have only presented the definition of *reputation* but we have not stressed the different types of reputation we consider in our approach. We distinguish between two different dimensions of reputation:

- *Reputation as a service provider*. This value represents the degree of satisfaction an agent has obtained after using a service provided by another agent, i.e., it indicates the quality of the service provided in the point of view of the agent using such service. This reputation reflects the fulfillment and violations of organizational norms ($\mathcal{ON}$) in an interaction playing a specific role.
- *Reputation as a reputation source*. This kind of reputation is related to the degree of satisfaction that an agent obtains after requesting opinions about others to a third party. The reputation of an agent as a reputation source evaluates the behavior of such agent while providing information about the reputation of others. If an agent has a bad reputation as *reputation source*, the opinions it has provided about other agents should not be trustworthy. The reputation of an agent as a reputation source is related to the role *reputation provider* and to a unique norm that prohibits the agent from lying when informing others about a counterpart's reputation.

It is fundamental to distinguish these two different dimensions of reputation since, on the one hand, the former deals with the quality, competence, availability, etc. of the agent which is requested for some kind of interaction to provide a service. On the other hand, the second reputation is calculated in order to measure how popular and accurate is another agent providing reputation information about third parties[1]. Both values are important depending on the information the agent needs in each moment.

2.4 Organizational and Individual Norms

At the beginning of this section we have defined an organization as an entity which is formed, among others, by a set of organizational norms ($\mathcal{ON}$) that regulate the behavior of agents in different situations.

[1] Reputation as a reputation source could be seen as a particular case of the first type of reputation. It is the agent's reputation when providing the service of informing others about reputation of another one. We distinguish both types in order to make clear the different natures they have and the different ways of assess them.

Definition 5. *Let* $\mathcal{ON}_{\mathcal{O}_o}(\mathcal{R}_j, \mathcal{I}_k)$ *be an* organizational norm *of the organization* $\mathcal{O}_o \in \mathcal{O}$ *applied to the members of the organization playing the role* $\mathcal{R}_j \in \mathcal{R}$ *in the interaction* $\mathcal{I}_k \in \mathcal{I}$*. If* $\mathcal{R}_j$ *is not specified, the norms are applied to all agents playing any role in the interaction* $\mathcal{I}_k$*. If* $\mathcal{I}_k$ *is not defined, the norms are applied to all agents playing the role* $\mathcal{R}_j$ *in any interaction. If neither* $\mathcal{R}_j$ *nor* $\mathcal{I}_k$ *are specified, the norms are applied to all members of the organization* $\mathcal{O}_o$*.*

Note that when describing an (il)legal situation, the norm being mentioned in the situation must be a valid norm, i.e., must be a norm that is defined for the role being mentioned and for the interaction being described. From a social point of view - *macro level* - $\mathcal{ON}$ are global norms commonly accepted by all organization members. They are imposed by the organization and are publicly advertised. On the other hand, from an individual point of view - *micro level* - agents may define their own norms by specializing the organization norms and making them more restrict. Such norms, called *individual norms*, reflect the relevance the global norms represent to the agents and cannot be used as a mechanism to modify or delete organization norms. In contrast to organizational norms, individual norms are not public but can be shared with anyone according to the agent decision. While evaluating the behavior of agents, an agent may consider both the organizational and individual norms. An individual norm that specializes an organizational norm $\mathcal{ON}_x$ is also related to the same $(\mathcal{R}_j, \mathcal{I}_k)$ tuple defined by $\mathcal{ON}_x$, that is, the individual norm is applied to the same interaction and role as the organizational norm that it specializes.

Definition 6. *Let* $\mathcal{IN}_{\mathcal{A}_i}(\mathcal{R}_j, \mathcal{I}_k)$ *be an* individual norm *applied by agent* $\mathcal{A}_i$ *to the agents playing the role* $\mathcal{R}_j \in \mathcal{R}$ *in the interaction* $\mathcal{I}_k \in \mathcal{I}$*.*

The reputation $Rep_{\mathcal{A}_i \to \mathcal{A}_j}$ resulting from applying agent $\mathcal{A}_i$'s assessment about other agent $\mathcal{A}_j$ regarding to violations and fulfillment of norms is as follows:

- If $\mathcal{A}_j$ has fulfilled the organizational norm[2], then $Rep_{\mathcal{A}_i \to \mathcal{A}_j} \in [0, +1]$. The specific value will be generated by the agent according to the fulfillments of the individual norms. If no individual norm has been fulfilled then $Rep_{\mathcal{A}_i \to \mathcal{A}_j} = 0$.
- If the organizational norm has been fulfilled and so have the individual ones, then $Rep_{\mathcal{A}_i \to \mathcal{A}_j} = +1$.
- On the other hand, the violation of the organizational norm entails that $Rep_{\mathcal{A}_i \to \mathcal{A}_j} \in [-1, 0)$. The adjustment of the specific value calculated in this range will result from checking individual norms.
- If the organizational norm and the individual norms have been violated, then $Rep_{\mathcal{A}_i \to \mathcal{A}_j} = -1$.

By using this approach we allow agents, on the one hand, to specify at their will the reputation values to be associated with the agents behavior, and, on the other hand, to use a one-step argumentation when sending all the information related to a reputation value to others (included in the organization).

[2] Here the organizational norm refers to that from which the individual norm is specialized.

3 Decentralized Mechanism

In order to make use of our proposed reputation mechanism, the agents must be able to individually evaluate (as well as storing) the behavior of other agents with whom they have interacted and also of providing such information to the organization they belong to (centralized part) or to other agents if requested. Such evaluation must be made according to the set of organizational norms defined in the organization and also according to the set of individual norms defined by the agent itself.

3.1 Evaluating Agents' Behavior When Providing a Service

We are assuming that agents are able to interpret norms and also to identify that an action is violating a norm. In order to help agents in doing such tasks, approaches such as [10], can be used.

While evaluating the behavior of other agents, an agent must focus on the service being provided, i.e., on the interaction being checked and on the role being played by the other agent. A reputation value should be generated for each $\langle \mathcal{R}_j, \mathcal{I}_k \rangle$ tuple by separately considering each norm that regulates such interaction and/or role. Note that there may be norms applied to all agents regardless of the role being played and the interaction where the agent is participating. Those norms must also be considered.

An agent should start its evaluation by checking if the organizational norms have been fulfilled. The agent must check if the other agent has fulfilled the obligations and has not violated the prohibitions defined by the organization. The organizational norms are public to all agents executing in the organization. Every time an agent enters in the organization, it is informed about *i)* the norms to be fulfilled related to the role to be played, and *ii)* other norms applied to all agents in the organization regardless of the role they play. Agents must request for both types of norms to the organization before starting any interaction.

After verifying if the organizational norms have been fulfilled or violated[3], the agent may verify if its individual norms were fulfilled, in the case they have been defined. The individual norms to be analyzed are those applied to the same $\langle \mathcal{R}_j, \mathcal{I}_k \rangle$ tuple and also those that do not dependent on the role or interaction being analyzed.

To illustrate such idea, consider the following example. Let's suppose that *Alice* playing role *Flight Customer* has made a reservation of a flight ticket and that *Bob*, playing the role *Flight Provider*, has cancelled such reservation since the deadline for paying it has expired. The organization *Travel Agency* where both agents are involved has defined an organizational norm $\mathcal{ON}_1$ stating that *flight providers* can only cancel a reservation after the deadline for paying it.

$$\mathcal{ON}_1 : \text{PROHIBITION FlightProvider EXECUTE ticket.cancelTicket IF ticket.deadlineForPaying} \leq \text{TODAY}$$

[3] Note that verifying if a norm has been fulfilled is an objective action, while the interpretation of a violation of a norm - calculus of reputation - is clearly subjective.

If *Bob* cancels the flight after the deadline for paying the reservation, he has not violated the organizational norm and its reputation will thus vary from $[0, +1]$. However, *Alice* can still be dissatisfied. *Alice* expects *Bob* to cancel the reservation only 5 days after the deadline, described by the individual norm $\mathcal{IN}_{Alice_1}$.

$\mathcal{IN}_{Alice_1}$ PROHIBITION FlightProvider EXECUTE ticket.cancelTicket IF ticket.deadlineForPaying+5 $\leq$ TODAY

Such expectation characterizes an individual norm defined by *Alice*. If *Bob* cancels the reservation only 1 day after the deadline, he has violated such individual norm and, therefore, his reputation will be automatically lower than $+1$.

In order to be able to fulfil the individual norms defined by *Alice*, *Bob* can ask her about them. Note that we consider that all individual norms defined by *Alice* are specializations of organizational norms define by the *Travel Agency* organization, since they are more restricted.

3.2 Evaluating Agents' Behavior as Reputation Sources

The behavior of agents while providing information about the reputation of other agents is regulated by only one organizational norm. Such norm states that the agent playing role *ReputationSource* in the interaction *ReputationInformationExchange* cannot lie while providing that information. The norm that regulates such behavior is described by $\mathcal{ON}$(ReputationSource, ReputationInformationExchange)[4]:

$\mathcal{ON}(ReputationSource, ReputationInformationExchange)$: PROHIBITION ReputationSource EXECUTE sendingWrongInformation

In contrast with other organizational norms, this is the unique norm that is domain-independent since it neither depends on the roles nor interactions defined in the organization.

The reputation of the agent as a *reputation source* can only be evaluated after the agent that is receiving the information has interacted with the desired agent (the third party it requested for). After the interaction, the agent that has received the information is able to evaluate if the reputation value it has received is consistent with the counterpart's behavior (from agent's individual view). This evaluation will form the reputation value for the agent that played the role *ReputationSource*.

Let's suppose *Alice* has never interacted with *Bob* before; then she asks *Carol* about how "good" is *Bob* playing role *Flight Customer*. After interacting with *Bob*, *Alice* evaluates his behavior and compares it with the information she has received from *Carol*. In the case the reputation values are similar, the reputation of *Carol* as a reputation source will be good, i.e., will be in the range $[0, +1]$. In

[4] Although we are not describing sanctions to the norms used in this paper as examples, punishments and rewards can be specified.

the other case, *Carol*'s reputation as reputation source will be bad from *Alice*'s point of view, i.e., will be in the range $[-1, 0)$. The specific reputation value will depend on how different the reputation being informed is from the reputation being evaluated by the agent herself. To present the equation to be used by the agent is out of the scope of the paper, since we consider this is a function every agent should define by itself.

3.3 Sending and Receiving Agents' Reputations

Our proposed decentralized mechanism makes possible the sharing of reputation values between the agents. Agents can send and receive reputation values by using the *reputation information* ($\mathcal{RI}$) tuple defined in 2.2. Such information is composed by the following contents $\langle \mathcal{P}_q, ((\mathcal{A}_i, \mathcal{R}_j, \mathcal{I}_k, t), \mathcal{N}_m, \mathcal{F}_n), Rep_{\mathcal{P}_q \rightarrow \mathcal{A}_i} \rangle$:

- *Provider* ($\mathcal{P}_q$): the agent providing the evaluation about another agent's behavior;
- *(Il)legal Situation*: a legal or illegal situation evaluated according to a norm:
  - *Situation*: the situation being evaluated;
    - *Client* ($\mathcal{A}_i$): the agent whose behavior is being evaluated;
    - *Client role* ($\mathcal{R}_j$): the role that the client was playing and that is related to the behavior being evaluated;
    - *Interaction* ($\mathcal{I}_k$): the relation between the provider and the client associated with the evaluation;
    - *Time* (t): the time where the situation has occurred;
  - *Norm* ($\mathcal{N}_m$): the organizational or individual norm that is being considered. It can be a violated or fulfilled norm;
  - *Fact* ($\mathcal{F}_n$): the action(s) that was(were) executed (that is fulfilling the norm, in the case of an obligation, or violating the norm, in the case of a prohibition) or the action(s) that was(were) not executed (that is fulfilling the norm, in the case of an prohibition, or violating the norm, in the case of an obligation) during the interaction.
- *Reputation* ($Rep_{\mathcal{P}_q \rightarrow \mathcal{A}_i}$): a value that represents the evaluation about the agent's behavior;

In the case of reputations related to reputation sources, it is not necessary to state the norm since there is only one norm associated with such value, as stated in section 3.2. It is also not necessary to state the client role and interaction since it is always the same interaction type.

4 Centralized Mechanism

As we have introduced in section 2, our centralized mechanism should be implemented in organizations and, thus, be common to all participants in such organizations. The main objective of the mechanism is to receive the agents' past experiences as reputation values together with the information used to justify such value. The mechanism puts together such information and can provide

it to any agent, including to new ones. The main advantage of our proposed centralized mechanism is its ability of pointing out the violated and fulfilled norms and the facts related to such violations and fulfillments while informing the agents' reputations. In addition, the mechanism is also able to inform who are the agents that have provided the information and also to inform about their reputations as reputation sources.

4.1 Sending and Receiving Agents' Reputations

Every time an agent evaluates the behavior of another one, it may send such evaluation as *reputation information ($\mathcal{RI}$)* to the organization. The organization is able to put together all reputations and provide them to any other agent.[5] As a passive entity, the organization aggregates $\mathcal{RI}$ tuples and makes rankings using such information. Therefore, agents cannot call the organization as an entity capable of lying, since it only acts as a means for publicizing information.

In order to illustrate that idea we show some examples of *reputation information* provided by the organization using the format of the tuple $\mathcal{RI}$: $\langle \mathcal{P}_q, ((\mathcal{A}_i, \mathcal{R}_j, \mathcal{I}_k, t), \mathcal{N}_m, \mathcal{F}_n), Rep_{\mathcal{P}_q \to \mathcal{A}_i} \rangle$

- Overall reputation of an agent $\mathcal{A}_i$ as a service provider from the agent $\mathcal{P}_q$ point of view. It gives a value representing aggregated opinions provided by $\mathcal{P}_q$ about the agent $\mathcal{A}_i$ in all situations in which it has participated regardless of the norms it has violated or fulfilled.

$$\langle \mathcal{P}_q, ((\mathcal{A}_i, \_, \_, \_), \_, \mathcal{F}_n), Rep_{\mathcal{P}_q \to \mathcal{A}_i} \rangle$$

- Reputation of an agent $\mathcal{A}_i$ playing the role $\mathcal{R}_j$ from the point of view of the agent provider $\mathcal{P}_q$. All reputation values provided by $\mathcal{P}_q$ related to all situations where the agent $\mathcal{A}_i$ is playing the role $\mathcal{R}_j$ are grouped.

$$\langle \mathcal{P}_q, ((\mathcal{A}_i, \mathcal{R}_j, \_, \_), \_, \mathcal{F}_n), Rep_{\mathcal{P}_q \to \mathcal{A}_i} \rangle$$

- Reputation of an agent $\mathcal{A}_i$ while providing a specific service from the point of view of the agent provider $\mathcal{P}_q$. It gives the reputation of the agent in a given situation, i.e., while playing a role $\mathcal{R}_j$ in an interaction $\mathcal{I}_k$. Such reputation value represents how trustworthy an agent is while providing a service from the point of view of the agent $\mathcal{P}_q$.

$$\langle \mathcal{P}_q, ((\mathcal{A}_i, \mathcal{R}_j, \mathcal{I}_k, \_), \_, \mathcal{F}_n), Rep_{\mathcal{P}_q \to \mathcal{A}_i} \rangle$$

- Overall reputation of an agent $\mathcal{A}_i$ as reputation source from the point of view of the agent provider $\mathcal{P}_q$. It gives a value representing all opinions provided by $\mathcal{P}_q$ about the agent $\mathcal{A}_i$ while providing reputation values about other agents behavior, i. e. how the agent $\mathcal{A}_i$ is regarding to the organizational norm $\mathcal{ON}_l$ which describes that an agent cannot lie when providing reputation information.

$$\langle \mathcal{P}_q, ((\mathcal{A}_i, \_, \_, \_), \mathcal{ON}_l, \mathcal{F}_n), Rep_{\mathcal{P}_q \to \mathcal{A}_i} \rangle$$

[5] Note that organizations must not modify the information that they receive.

– Also, the organization could provide rankings[6] such as the best agents providing any kind of service, the best agents being reputation sources, the agents with the highest *social prestige* (see section 4.2), the most collaborative agents based on the agents which send more opinions to the organization, etc.

Note that organizations are not limited to provide only the above mentioned services, other services can also be included. In addition, organizations can implement different algorithms to provide this information. For example, some can only consider the most recent reputations while others can consider all reputations they have received over time.

4.2 Agents Motivations

An agent can contribute to the organization by providing information about other agents as services providers and by providing information about other agents as reputation sources. The main goal of an agent on sending such information is to increase its *social prestige*, i.e., the image that the agent can offer about itself. The *social prestige* of an agent is increased when it advertises other agents about the ones they can trust as services providers. Such information helps then on distinguishing trustworthy agents from others. Its *social prestige* also increases when the agent provides information about the others that can be trusted as reputation sources. Such information helps the agents on distinguishing the ones that are sending trustful information about the services provided by other agents[7].

In order to make public the *social prestige* of the agents, the organization provides a ranking of the best agents on sending trustful informations to the organizations (section 4.1). Since such ranking can be used by agents while selecting the ones to interact with, the definition of the concept of *social prestige* in our model motivates the agents to provide information to the organization.

5 Discussion

Our proposed approach is based on the benefits provided by the two most common mechanisms used for implementing reputation systems while trying to solve some of their drawbacks. The advantages of centralized mechanisms such as [2,3] are *i)* the availability of the reputations to any agent in the system and *ii)* the use of global reputation values. Agents that have recently joined the organization or agents that have been playing roles can ask the centralized mechanism for reputations of other agents. It is not necessary to search for agents that could provide such information - since it is available in the organization -, or to

[6] All rankings are built based on the opinions received by the agents.

[7] In order to avoid cheating agents increase their social prestige at the expense of others, agents should implement their aggregation functions - for received values - as an average value weighted by the reputation of the reputation source.

ask for different testimonies in order to decrease the risk of misunderstanding - since the organization aggregates all provided reputation information about every agent. Some drawbacks of centralized mechanisms are ($C1$) they simply put the reputation values together - there is no reasoning about them, ($C2$) they may not be scalable, ($C3$) the agents that has provided the information might be unknown and ($C4$) there is not any information about what has influenced the agent reputation value.

Trying to solve the problems pointed out in $C2$ and $C3$, some authors have proposed the use of decentralized mechanisms [1,9,15,16]. However, those mechanisms had a serious problem: ($D0$) in order to know the reputation of a possible future partner, the agent should search for other agents that had previously interacted with the former, so providing their testimonies about such interactions. In large-scale MAS we can consider that the process of searching for those agents may be difficult and take a lot of time.

This problem was solved in decentralized mechanism that have adopted certified reputations [4,7,12]. Each agent has numeral certified reputations (or references) provided by the agents they have interacted with and, thus, can offer those references to the agents that intend to interact with it. The main advantages of those decentralized mechanism are *i)* the high availability of the reputations since any agent can easily learn the reputation of any other, *ii)* the fact that reputations are individually stored making the solution scalable and *iii)* the knowledge about the source agents that are providing the reputation values. The main disadvantages of such approaches are ($D1$) it may be difficult to find the agents that are trustworthy as reputation sources, ($D2$) agents must meet frequently in order to establish strong links among them and to pass on consistent reputations and ($D3$) there is not any information about what has influenced the agent reputation value.

In order to overcome some of those problems, the hybrid reputation system presented in [11] provides a centralized mechanism that may store the recently reputation values (solving problem $C2$) where agents can ask for reputations of other agents provided by several agents in different experiences (solving problem $D2$ since one single agent does not need to meet frequently with other agents to be able to have a consistence evaluation of their behavior). In addition, the agents executing in this system are also able to evaluate the past behavior of other agents and store the reputation values. Different from the centralized approaches, the proposed centralized mechanism groups the reputations according to the violated norms and the roles being played (solving problem $C1$). The mechanism is able to provide the reputation of an agent considering a specific norm or one of the roles of the agent. But it is not able to provide information about the facts that characterize the violations and about the agents that have provided the reputation values (it cannot solve problems $C3$, $C4$, $D1$ and $D3$). Since agents make a personal evaluation about the behavior of other agents, it is fundamental to know the facts that violated the norms in order to interpret the reputation values.

The hybrid reputation model proposed in this paper solves the problems pointed out in this section. The agents using the decentralized mechanism are

able to evaluate the behavior of other agents, to store such evaluation (contributing to solve problem $C2$ since the centralized mechanism will probably store only the recent reputation information) and to provide to the organizations where they are involved information about their reputations, norms violated and fulfilled and facts violating and fulfilling the norms (solving problem $D3$). The organizations implement the centralized mechanisms that are able to store the (recent) reputation values (solving problem $C2$), the information related to such values (solving problem $C4$) and the identification of the agents that have provided them (solving problem $C3$). The centralized mechanism is able to provide such information (grouped in many different ways) to any agent in the system (solving problems $C1$ and $D2$). The reputations are evaluated according to the norms that regulate a situation (role + interaction) where the agent being evaluated is involved. The model distinguishes between two different types of reputation: reputation as *service providers* and reputation as *reputation sources*. Therefore, it is easy to know the agents that are trustworthy as reputation sources (solving problem $D1$).

6 Conclusions

In this paper we have presented a hybrid reputation model for organizations which try to solve problems usually associated with centralized and decentralized reputation mechanisms.

The main advantages of our proposed model are: *i)* the reputation values are exchanged together with the violations and fulfillments of norms and the facts related which those violations or fulfillments (as a single-step argumentation process); *ii)* the organizations are able to provide non-manipulated information about the reputations of agents; *iii)* our proposed model distinguishes between two types of reputation: *reputation as a service provider* and *reputation as a reputation source*; and *iv)* the model motivates agents on sending information to the organization by making available a ranking stating the agents with highest *social prestige*.

As future work we plan to test our reputation model in some organizational domain such as travel agencies. We also intend to compare our model with other centralized or decentralized reputation models to observe the advantages of the hybrid proposals, as well as with other hybrid models to observe the benefits of our model opposite to others. Finally, we want to integrate our proposal in some proposed trust model in order to observe how it behaves working together with a confidence model.

References

1. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 6, pp. 293–301 (2000)
2. Amazon (2008), `http://www.amazon.com`

3. eBay (2008), http://www.ebay.com
4. Grandison, T., Sloman, M.: A survey of trust in internet applications. IEEE Communications Surveys & Tutorials 3(4) (2000)
5. Hermoso, R., Billhardt, H., Centeno, R., Ossowski, S.: Effective use of organisational abstractions for confidence models. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 368–383. Springer, Heidelberg (2007)
6. Hermoso, R., Billhardt, H., Ossowski, S.: Integrating trust in virtual organisations. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 17–29. Springer, Heidelberg (2007)
7. Huynh, T., Jennings, N., Shadbolt, N.: Fire: An integrated trust and reputation model for open multi-agent systems. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), pp. 18–22 (2004)
8. Omicini, A., Ossowski, S.: Objective versus subjective coordination in the engineering of agent systems. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) Intelligent Information Agents. LNCS (LNAI), vol. 2586, pp. 179–202. Springer, Heidelberg (2003)
9. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of First International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 475–482 (2002)
10. Silva, V.: Implementing norms that govern non-dialogical actions. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 232–244. Springer, Heidelberg (2008)
11. Silva, V., Duran, F., Guedes, J., Lucena, C.: Governing multi-agent systems. Journal of Brazilian Computer Science 2(13), 19–34 (2007)
12. Skogsrud, H., Benatallah, B., Casati, F.: Model-driven trust negotiation for web services. IEEE Internet Computing 7(6), 45–52 (2003)
13. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.P.M.: Implementing norms in multiagent systems. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 313–327. Springer, Heidelberg (2004)
14. López y López, F.: Social power and norms: Impact on agent behavior. PhD thesis, Univ. of Southampton, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science (2003)
15. Yu, B., Singh, M.: Distributed reputation management for electronic commerce. Computational Intelligence 18(4), 535–549 (2002)
16. Yu, B., Singh, M.: An evidential model of distributed reputation management. In: Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems, vol. 1, pp. 294–301 (2002)

Formalising Situatedness and Adaptation in Electronic Institutions

Jordi Campos[1], Maite López-Sánchez[1], J.A. Rodríguez-Aguilar[2], and Marc Esteva[2]

[1] MAiA Deptartment, Universitat de Barcelona
{jcampos,maite}@maia.ub.es
[2] Artificial Intelligence Research Institute (IIIA), CSIC
{jar,marc}@iiia.csic.es

Abstract. Similarly to institutions in human societies, an Electronic Institution (EI) provides a structured framework for a Multi-Agent System (MAS) to regulate agents' interactions. However, current EIs cannot regulate a previously existing dynamic social system and deal with its agent population behaviour changes. This paper suggests a solution consisting of two EI extensions to incorporate situatedness and adaptation to the institution. These two properties are usually present at an agent level, but this paper studies how to bring them to an organisational level. While exposing our approach, we use a traffic scenario example to illustrate its concepts.

1 Introduction

Historically, societies have been organised based on conventions that individuals conform and expect others to conform [1]. Within organisations, conventions are explicit and are stated in terms of rules, protocols or both. In the context of Multi-Agent Systems (MAS), Electronic Institutions [2] try to follow the same principles.

An Electronic Institution (EI) is a MAS framework designed to guarantee previously defined social conventions. These conventions are designed to let the whole system achieve certain implicit goals. For instance, traffic rules try to improve traffic flow and to avoid accidents. These social conventions support agent coordination which typically has been handled from two main approaches [3][4][5]: considering the individual perspective of agents –so conventions can emerge– or a global organisational perspective —using infrastructure to support them. An EI provides an organisational approach that regulates the agent interaction, thus the institution follows a global coordination perspective instead of an individual approach. More concretely, an EI is a *self-contained* and *static* organisational framework. By *self-contained* we mean agent interaction solely occurs inside the institution. Thus, we assume an EI mediates all messages among its participants. Also, an EI's social conventions do not change during its execution, so we consider it is *static*.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 126–139, 2009.

A problem arises when we have a dynamic social system –we call it *world*– and we want to enhance it by adding an EI: the fact of being *self-contained* and *static* becomes a limitation. Instead of being *self-contained*, we need the institution to be aware of the *world* it is added to. Even more, the institution should adapt to changes of this dynamic social system. In this manner, we pursue the enhancement of the existing dynamic social system by addition of EI's regulating capabilities.

Our proposed solution to cope with this limitation is to extend an EI into a *situated autonomic* organisational framework. By *situated* we mean it is aware and can induce changes in the external social system (it is bound to *world*). And by *autonomic* we mean it can autonomously adapt to changes in the dynamic existing social system. We envisage the EI as a whole, *autonomic* and *situated* in a *world*. This vision at organisational level is very similar to autonomous situated agents at individual level. Thus, we conceive our proposed extension to EI as bringing to an organisational level two agent properties: situatedness and adaptation.

Specifically, we consider the institution situatedness as an awareness of its *world* (society, organisation or MAS) and its capacity to induce changes on it. In this paper, we formalise and extend some concepts used in a previous approach [6] to situate an EI. Besides, we envision adaptation as a goal-driven mechanism to change conventions. Societal changes, such as changes in agent behaviours or properties, may affect negatively in the fulfilment of organisational goals. Just as agents must adapt in order to succeed, Electronic Institutions should be able to adapt to fulfil their own global goals —which may differ from individual ones. Thus, an extended institution is able to modify its conventions to improve the system's effectiveness to accomplish the organisational goals —it may also improve the system's efficiency. This adaptation can also be seen as a reconfiguration aspect of autonomic computing, where systems are able to reconfigure themselves without human intervention [7]. In this paper, we formalise and extend some concepts used in previous approaches to situate an EI [6] and to adapt it [8].

Along this paper we use a traffic scenario as an example to illustrate introduced concepts (see Figure 2). In this scenario, an Electronic Institution acts as a *Traffic Regulation Authority*. Most agents play the role of cars, but we also consider policemen agents which act on behalf of the institution. These agents interact in a two-road junction, each road having two lanes in opposite directions. Lanes entering the junction have traffic lights controlled by our institution. When driving, cars enter and leave these crossroads at/from random sides. Moreover, cars may decide not to stop at red traffic lights, if this is the case and a policeman sees this traffic violation, it will sanction the car by subtracting points from its driving license. Finally, cars can collide. Collisions have an associated emergency protocol, in which a tow truck takes them from the crossroads to a garage to be repaired.

The rest of the paper is structured in five sections. Section 2 introduces Electronic Institutions to settle the basis for subsequent sections, which are devoted

to situatedness and adaptation. Section 3 presents the so-called *Situated Electronic Institution*, and section 4 defines the notion of *Autonomic Electronic Institution*. Next, both approaches are compared with their related work in section 5. Finally, section 6 exposes the conclusions and outlines paths to future research.

2 Electronic Institutions (EI)

An *Electronic Institution* (EI [2]) is an interaction framework for Multi-Agent Systems (MAS). One of the main objectives is to guarantee that its conventions –interaction protocols and rules– are followed by participant agents, which interact via dialogical actions. This is achieved by communication mediation, so that EIs filter out non-permitted actions. Figure 1 depicts this scheme. Participant agents are considered to be external to the institutional framework, and they interact through an institution wrapper called *governor*. Nevertheless, the institution delegates its functions to a special kind of agents, the so called *staff agents*. Accordingly, the definition of an EI is shown below and some of its components are discussed in next subsections:

Definition 1. *An Electronic Institution is a tuple* $EI = \langle DF, DC \rangle$ *[9]:*

- $DF = \langle O, M_I, ST, L_{CL}, L_E \rangle$ *stands for Dialogical Framework and provides a context for agent interactions, which are speech acts. Its components are: an Ontology* O*, a set of Information Models* M_I *–to keep information about EI's participants and activities at run time–, a Social structure* ST *–roles and their relationships–, a Communication Language* L_{CL} *–detailed in section 2.1–, and an Expression Language* L_E *—to specify conditions with a constraint language and their consequences in an action language.*
- $DC = \langle PS, NS \rangle$ *stands for a Deontological Component which is a set of conventions that constrains possible illocutionary exchanges and manages the responsibilities established within the institution. Its components are:* PS

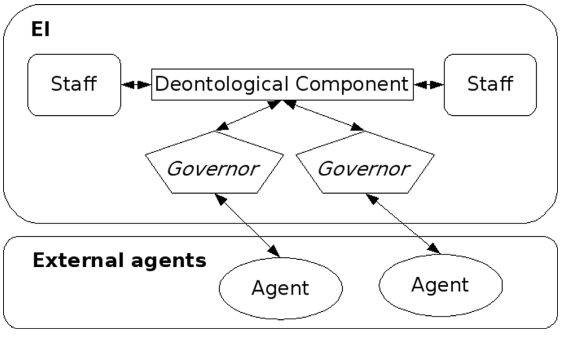


Fig. 1. Within an EI, participant agents interact through illocution messages mediated by their *Governors*. A Deontological Component and *Staff* agents guarantee the compliance of conventions.

as a Performative Structure and NS as a Normative Structure, both are described in subsequent sections.

2.1 Communication Language

The *Communication Language* (L_{CL}) is the language used by agents to utter their messages. Its expressions, called *illocutions* (I), are defined in terms of:

$$I ::= \iota(orgA_i : orgR_i, [dstA_j :] \; dstR_j, msg, t)$$

where there is an illocutionary particle ι (e.g. `request`, `accept`, `inform`...), its sender (an agent identifier $orgA_i$ and the role $orgR_i$ it plays), its receivers (an agent identifier $dstA_j$ or its role $dstR_j$), a message content $msg = f(params)$ and a time stamp t[1]. As an illustration, the following message could appear in the traffic scenario when police officer 'Bond' informs car 'Shiny' that it has a 10-point fine at time 1: `inform`(`Bond` : `policeman`, `Shiny` : `car`, `fine`(10), 1)

2.2 Performative Structure

A *Performative Structure* (PS) defines those conventions that regulate the flow of illocutions in an institution. The whole activity of an EI is a composition of multiple, concurrent dialogic activities –the so called scenes– involving different groups of agents playing different roles.

Each scene is specified by means of a finite-state directed graph, with nodes representing states and arcs defining those relevant actions that imply state transitions. It also includes some restrictions about time variables or how many agents can play a given role.

2.3 Normative Structure

The *Normative Structure* (NS) [10] defines a normative level in our Deontological Component. As described in [10] PS and NS are distributed and controlled by staff agents, called *Scene Managers* and *Normative Managers*. Briefly, a NS consists of a *Normative State* (S) and a set of *Rules* (R) that can update this state[2]:

$$\begin{aligned} NS \quad &= \langle\, S, R \,\rangle \\ S &= \{\, p_1 \ldots p_{n_S} \,\},\ p_i := \texttt{utt}(I) \mid NP,\ NP ::= \texttt{per}(I) \mid \texttt{prh}(I) \mid \texttt{obl}(I) \\ R &= \{\, r_1 \ldots r_{n_R} \,\},\ r_i \textit{ is a Rule} ::= Cond \Rightarrow Conseq \\ & \quad r_i : \mathbb{S} \times \mathbb{C}ond \rightarrow \mathbb{S} \\ & \quad Cond ::= \texttt{utt}(I) \mid NP \mid Cond, Cond \\ & \quad Conseq ::= \texttt{add}(NP) \mid \texttt{remove}(NP) \end{aligned}$$

[1] EIs have a distributed architecture assuming a synchronised time.

[2] We use uppercase letters to denote sets of elements (e.g. R is a set of rules) and lowercase letters to denote their elements (e.g. r_i is a single rule). In addition, when defining functions, we use blackboard letters to denote their domains (e.g. $\mathbb{S}$ is the domain of all possible *normative states* S).

Normative State (S) contains a set of statements called *Normative Positions* (NP), which represent obligations (`obl`), prohibitions (`prh`) and permissions (`per`) associated to illocutions (I). This state can be updated by agents utterances (`utt`) and *rules* (R) [11]. A *Rule* consists of a condition and its consequences. When it is triggered by any combination of uttered illocutions (`utt`) and NP, it adds or removes NPs to S. See section 3.2 for a normative example in the traffic scenario.

3 Situated Electronic Institutions

An Electronic Institution is an open MAS[3] that provides an interaction mediated environment within the institution itself, so we can refer to it as *EI inner environment*. In fact, an EI have total control over this *EI inner environment* thanks to its *governors*.

In this paper, we propose to extend an EI to be able to interact with a previously existing environment, so that we relax total control in favour of interoperability. We call this extended institution a *Situated Electronic Institution* (SEI). The existing environment –we call it *world*– can be any social system –society, organisation or MAS– having individual actions and interactions that are relevant to our institution. These actions and interactions in the *world* can be

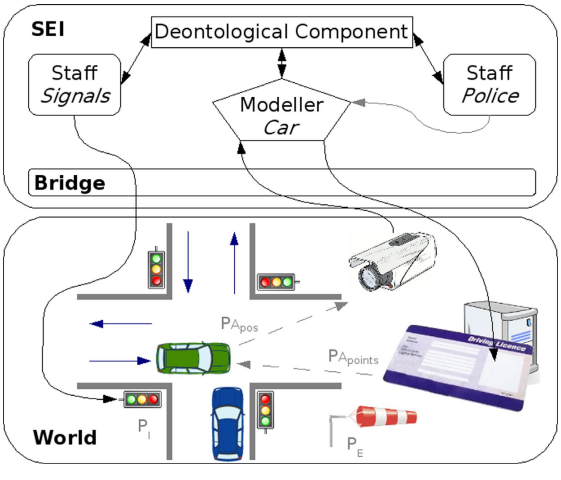


Fig. 2. A *Situated Electronic Institution* (SEI) in our traffic example. There is a communication *Bridge* between our SEI and the *World* it is situated in. This *bridge* allows *staff* agents to access certain *world* elements as *properties* (about modelled agents P_A, institutional issues P_I, or environment facts P_E). *Modellers* are specialised *governors* that model a *world* entity as if it was a regular external agent in an EI.

[3] By open MAS we mean systems populated by heterogeneous and self-interested agents, that are not known beforehand, may vary over time and can be both human and software agents developed by different parties. Hence, we can not expect participants to follow the social conventions established by an EI.

illocutions and non-verbal actions. The SEI-*world* relationship is accomplished by attaching a SEI on top of a *world* (see Figure 2). In this way, a SEI can perceive *world* facts and induce changes on it.

We say that a SEI is *situated* in this existing environment because it receives information about the environment, processes it, and induces some changes in the *world* to try to enhance its performance given some goals —they are implicit in protocol and rule definitions. We consider a SEI has a model of the *world*, which maintains –according to external inputs– and updates —translating changes to external environment. Similarly to agent level, at organisational level *world* may be also partially observable by a SEI. We also assume SEI's control over this *world* is quite limited, since it can only induce a limited amount of changes in it. Hence, a SEI can be defined as an extension of previous EI.

Definition 2. *A Situated Electronic Institution is a tuple* $SEI = \langle DF', DC, B \rangle$*:*

- DF' *stands for a previous Dialogical Framework extended with what we call* world*'s entity* Modellers *and* Properties*.*
- DC *corresponds to the Deontological Component of an EI.*
- B *stands for a Bridge, a communication channel with the world.*

3.1 Modellers and Staff Agents

Since a basic EI is a persistent SW framework, it uses its *Information Model* (M_I) to keep information about EI's computational state in the form of attributes. Now, in a SEI, we call *Agent Properties* ($P_A \subseteq M_I$) those attributes that keep the institutional state of each external agent (e.g. agent's credit or position), *Environment Properties* ($P_E \subseteq M_I$) to those attributes about global facts independent of the institution activity (e.g. date or weather) and *Institutional Properties* (P_I) to attributes related with global facts directly or indirectly influenced by the institution (e.g. the number of collisions which may be influenced by traffic lights' colours).

Some *external agents* of a SEI are represented by relevant *world* entities that are not controlled by the institution. Thus, a SEI has specialised *governors*, we name *Modellers*, in charge of modelling and interacting with these *world* entities. Thus, a *world* entity can be treated by the SEI as if it was a regular participant agent.

The information between SEI and *world* flows in two directions. On the one hand, a *Modeller* models a *world* entity by accessing the *world* and extracting relevant information about a certain entity. As a result, a *Modeller* keeps track of its corresponding entity *Agent properties* (P_A) and utters illocutions when its entity performs actions that are relevant to the institution. On the other hand, a *Modeller* translates interactions from SEI into changes in its *world* entity's *Agent properties* (P_A). Figure 2 illustrates this process in our traffic scenario. First, the "Car modeller" gets its car location ($P_{A_{pos}}$) by processing the camera information. If this car (c) is entering the road junction through a given lane ($lane_{id}$), the *modeller* generates the illocution '$\texttt{inform}\,(\texttt{c} : \texttt{car}, : \texttt{policeman}, \texttt{entryJunction}(\texttt{lane}_{\texttt{id}}), \texttt{t})$'. This illocution

informs all policemen in the 'Crossroads' scene that the modelled car has performed the *entryJunction* relevant action. Later, if *modeller* is asked to decrease car's driving license points ($P_{A_{points}}$, see section 3.2), it will contact the *Traffic Regulation Authority* to perform this operation.

We see the institution situatedness as an awareness of the *world* where it is situated. Thus, we consider a SEI is aware of its *world* in the sense that it models and affects it. However, its *world* may or may not be aware of this SEI, depending on the domain. Domains present some restrictions on which information can be accessed and/or updated, which determines the level of SEI-*world* interaction and awareness. For example, in our traffic scenario, if the car's position is retrieved with camera's image processing, this car may probably not be aware of the SEI. In contrast, if the car is equipped with a Global Positioning System sensor device and sends its position to the *Traffic Regulation Authority*, it may probably be aware of the existence of a surveillance system like the SEI.

On the other hand, there may be some *world* entities directly controlled by the institution. In this case, instead of a *Modeller*, a SEI has *staff agents* in charge of them. Figure 2 depicts an example in the traffic scenario. where a *staff agent* called "Signals" sends information to the *world* to set a traffic light colour (P_I). *Staff agents* can also interact with *Modellers* to access to *Agent Properties* (P_A) or read *Environment Properties* (P_E, e.g. the wind's direction).

3.2 Norms

We call *relevant actions* those actions –or interactions– in a SEI's external environment (*world*) that affect its institutional model. Consequently, a SEI perceives or induces these *relevant actions* and bind them to the *world*. Within *relevant actions*, we distinguish between: *allowed actions* –those that follow social conventions– and *non-allowed actions* —the rest of *relevant* actions.

Moreover, we use a *norm* to refer to a social convention regarding an agents' interaction. Thus, *allowed actions* are those that follow *norms*. Accordingly, we consider that a *norm* can be violated if agents do not follow its convention, that is, if agents perform *non-allowed actions*. On the other hand, we use *rule* to identify an expression that defines the consequences of agents' actions. Hence, we can use a *rule* to define the consequences of a *norm* violation.

In an EI, most social conventions are specified through protocols so that *governors* filter out those illocutions not following them (*non-allowed actions*). In this way, an EI grants no participant can violate these conventions. In contrast, a SEI does not have such control over the *world* since it cannot prevent *world* entities from performing actions (or interactions). Thus, when designing a SEI, we have to pay special attention to the fact that it cannot prevent participants from violating *norms*. Consequently, a SEI needs to specify the consequences of violating these conventions with rules added to its *Normative Structure* (NS, see section 2.3).

As an illustration, Figure 3 contains an example in our traffic scenario. First, it exposes a social convention (n) about respecting traffic lights. Next, taking into account that a SEI cannot hinder agents in performing non-allowed actions,

S.Conventions: $n = $ "cars cannot go through a red light"
Specification: $NS = \langle S_0 = \{\}, R = \{r\}\rangle$
$r = utt\,(\,inform(x, policeman, y, car, noStop(Tlight), t_i)\,)$
$\Rightarrow add\,(\,obl\,(\,inform(x, policeman, y, car, fine(5), t_{i+1})\,)\,)$
Execution: $S_1 = \{utt\,(\,inform(p, policeman, c, car, noStop(Tlight), t_1)\,)\}$
$S_2 = \{utt\,(\,inform(p, policeman, c, car, noStop(Tlight), t_1)\,),$
$obl\,(\,inform(p, policeman, c, car, fine(5), t_2)\,)\,\}$

Fig. 3. A *Normative Structure* example in a SEI situated in our traffic scenario

the institution's *Normative Specification* (NS) includes a *rule* r to define the consequences of violating this *norm*. This *rule* should say that "any car violating the norm will be fined". However, our example delegates violation judgements to *staff agents* ("Policemen", in this case). Therefore, the corresponding rule codifies the obligation of a "Policeman" to fine a car when it informs the car went through a red light. Finally, the example shows an execution case. It starts with the empty normative state S_0. Then, when a "Policeman" informs that a car has gone through a red traffic, the original normative state incorporates the corresponding illocution, resulting in S_1. Afterwards, a *Normative Manager* applies *rule* r by adding an obligation to normative state S_2.

3.3 Bridge

The *Bridge* (B) is an asynchronous bi-directional communication channel between our institution and the *world* (in [6] it was conceived as a channel connected to a multi-agent simulator). This channel is used by *Staff agents* and *Modellers* to obtain information from the external environment and to induce changes in *world* as explained previously. It provides access to manage *Agent*, *Institutional* and *Environment properties*.

Basically, this *Bridge* comes from an implementation requirement since it binds our SEI and its *world*. From an implementation perspective, although it is a single concept, it may be distributed among different APIs (Application Program Interfaces) to access different programming objects that interact with *world* elements.

4 Autonomic Electronic Institutions

The aim of an Electronic Institution is to guarantee that its defined protocols and rules are followed by its participant agents. These protocols and rules have been designed to pursue some implicit goals. However, as the profile of agents may differ among different populations, original protocols and rules may not lead to design goals. We can avoid this by extending EIs with an adaptation mechanism that allows institutions to adapt to these societal changes. Hence, we define an *Autonomic Electronic Institution* (AEI) as an electronic institution that can autonomously adapt to achieve a set of defined goals. We propose goal

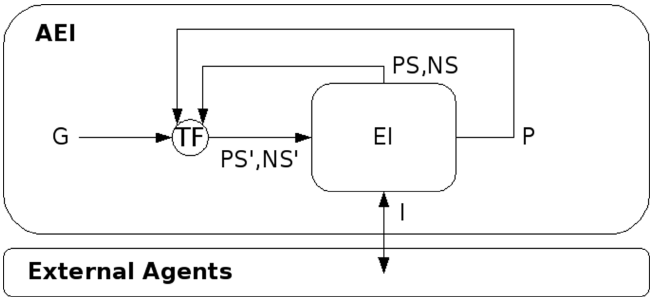


Fig. 4. An *Autonomic Electronic Institution* (AEI): *feedback* mechanism compares *observations* (prop. P) with their *expected values* (Goal G) and self-reconfigures (Perf.Struct. PS & Norm. Struct. NS) using Transition Func. (TF). External agents exchange illocutions (I) through the institution.

fulfilment to become the driving force for adaptation within the context of a rational world assumption. In this manner, an AEI has a *feedback* mechanism – centralised or distributed– with three main components: (1) an *objective* to define expected values of certain properties, (2) the corresponding observed *properties* and (3) a *mechanism* to specify how to reconfigure the institution to accomplish its objective depending on these observations (see Figure 4). Thus, we can define an AEI as an extension to an EI with these new elements.

Definition 3. *An Autonomic Electronic Institution is a tuple* $AEI = \langle DF, DC, G, TF \rangle$

- DF *and* DC *stand for a Dialogical Framework and Deontological Component*
- G *stands for institutional Goals*
- TF *stands for Transition Functions*

4.1 Institutional Goals

Institutional Goals (G) specify desired values for observed properties P. These properties belong to the information model ($P \subseteq M_I$), and correspond to information about agents, the environment or the institution itself (see section 3.1). Goals have the following components: $G = \langle GS, \Gamma \rangle$

$$GS = \{\, gs_{P_1} \dots gs_{P_{|GS|}} \,\},\ gs_{P_i} = \langle range_{P_i}, \gamma_{P_i} \rangle, \gamma_{P_i} : \mathbb{P} \rightarrow \mathbb{R} \in [0..1]$$
$$\Gamma \quad : \mathbb{GS} \times \mathbb{P} \rightarrow \mathbb{R} \in [0..1]$$

- *Goal Specifications* (GS): is a set of goal specifications over each observed property (P). Each goal specification (gs_{P_i}) is a definition of a property value expected range ($range_{P_i}$) and a function that evaluates its fulfilment grade (γ_{P_i}). This grade is a normalised real value between 0 and 1, being 1 the completely satisfied grade. In our *Traffic Scenario* GS tries to keep the number of norm violations below ten ($0 \leq P_{I_v} \leq 10$) and a minimum number of policemen ($0 \leq P_{I_p} \leq 1$).

– *Objective Function* (Γ): function that computes overall goal satisfaction (a real value between 0 and 1, 1 meaning completely satisfied goals) from defined goals and current observations. Following our example, we would get maximum goal satisfaction having no violations while no policemen are deployed in our traffic scene with a weighted aggregation function [12] (it is an utopian situation, however).

4.2 Transition Functions

Transition Functions (TF) specify how the institution can change its organisational structure with the aim of increasing its overall goal satisfaction. Our approach is that the institution contains one or more staff agents in charge of the adaptation (*Adaptation Managers*[4]). These staff agents reason following these transition functions given the observations and goals, and induce the changes in the institution according to the decided adaptation measure. We define two different transition functions depending on what they can adapt[5]. All of them receive a set of observed properties (P) and their expected values (e.g., institutional goal G). These properties can be any of the attributes described in section 3.1.

– *Normative Structure adaptation* ($\nu : \mathbb{P} \times \mathbb{G} \times \mathbb{NS} \rightarrow \mathbb{NS}$): it is a function in charge of updating rules (NS) if current observed properties (P) differ from expected values (G). In our traffic example, fines increase if there are a lot of traffic violations. *Normative structure* (NS) will be updated (NS'), by increasing the fine parameter (e.g. from 5 to 10) of rule r_a (see section 3.2):

$$
\begin{aligned}
NS &= \langle\, S, R = \{r_a\} \,\rangle \\
&\quad r_a : utt\,(\, inform(x, policeman, y, car, noStop(Tlight), t_i)\,) \\
&\qquad \Rightarrow add\,(\, obl\,(\, inform(x, policeman, y, car, fine(\mathbf{5}), t_{i+1})\,)\,) \\
NS' &= \langle\, S, R' = \{r'_a\} \,\rangle \\
&\quad r'_a : utt\,(\, inform(x, policeman, y, car, noStop(Tlight), t_i)\,) \\
&\qquad \Rightarrow add\,(\, obl\,(\, inform(x, policeman, y, car, fine(\mathbf{10}), t_{i+1})\,)\,)
\end{aligned}
$$

– *Performative Structure adaptation* ($\psi : \mathbb{P} \times \mathbb{G} \times \mathbb{PS} \rightarrow \mathbb{PS}$): it is a function in charge of updating protocols and/or role flows (PS) if current observed properties (P) differ from expected values (G). For example, a possible PS

[4] The distribution of the adaptation mechanism is out of the scope of this paper. Nonetheless, we think it would have two main axis: task decomposition (e.g. having an agent in charge of each adaptable norm or scene) and goal decomposition (e.g. distributed planning).

[5] Although this paper takes a formal approach, in a related work with a similar scenario [8], we study how these transition functions can be learnt if it is not possible to define them in advance. There, we apply a Genetic Algorithm (GA) technique to evaluate goal satisfaction with specific rules (NS) for a given participant population. Afterwards, rules (NS') of best GA individuals are stored in a Case-Based Reasoning (CBR) system, which substitutes the NS adaptation function.

adaptation is to update the number of agents playing a given role allowed in a certain scene (see section 2.2). Thus, in our traffic scenario, if there are a lot of accidents, function ψ would change the number of allowed policemen deployed in our 'Crossroads' scene.

5 Related Work

Multi-Agent System (MAS) approaches can be viewed [3][5] as *agent centred* or *organisation centred*. In general, previous work follows an *agent centred* approach [4]. However, the aim of this paper is to study two common individual agent properties –situatedness and adaptation– at an organisation level. In order to do it, we extend the notion of *Electronic Institution* (EI), which already is an *organisation centred* MAS.

The closest approach –that also uses an organisational approach– is $\mathcal{M}$OISE^{Inst} [13]. It provides a mechanism similar to EI *Governors* called *Wrapper Manager*. Hence, each agent uses its own *Wrapper Manager* to communicate with *OrgManager*. This *OrgManager* only changes the *Organisational Entity* state if *Wrapper Manager* petition does not violate any organisational constraint. Thus, we can establish several equivalences with EIs: *OrgManager* deals with a *ContextManager* that is equivalent to EI's *Scene Managers*; *Organisational Entity* states can be mapped into EI's *Information Model*; and the organisational constraints of their *Organisational Specification* correspond to our *scene* protocols. In addition, they also have a normative layer. Instead of extending original organisational framework with a new component –like EI's *Normative Structure*–, they conceive a *Global Normative Organisation* composed by two organisations: a domain one –the original organisation– plus a supervision one —a new organisation to supervise the original one. And all supervision organisation roles have *authority* to control domain organisation roles. Hence, they reuse mechanisms they already had defined instead of adding new ones. Given $\mathcal{M}$OISE^{Inst}-EI similarity, applying SEI and AEI's extensions to $\mathcal{M}$OISE^{Inst} seems feasible.

Regarding the concept of *situatedness* at organisation level, most of literature interprets it as providing a location notion to MAS participants. This idea was introduced by Weyns et. al. [14] as a way to allow local synchronisation of agents in the first *Situated MAS* approach [15]. The key point is to restrict participants' perceptions depending on their virtual location. CArtAgO [16] is also an example of this *perception paradigm*. It provides direct interaction among agents, and also indirect interaction through *artifacts*. But, in both cases, the scope of these interactions is limited to *workspaces* where these elements are located. It uses an *agent body* to situate an agent inside those *workspaces*; then its location determines which *artifacts* can be perceived or manipulated by its corresponding agent. In this sense, EI's *scenes* can be regarded as a way of grouping agents that can interact together, which can be interpreted as a virtual location that restricts their perception. *EASI* model [17] goes a step further, and additionally lets agents determine which element they want to perceive. They sustain this approach exposing that *awareness* is an active state. Precisely, we see EI's

situatedness as an *awareness* of the *world* where it is situated. A SEI, as a whole, determines its *world* perception and interaction. A first approach to this EI's *situatedness* was the *Simulator Bridge* [6]. However, we go further by assuming all external agents' interactions are performed in the *world*. A similar approach is detailed in [18], where they explore the idea of controlling physical entities with a MAS. They perform a global overview, without detailing changes in EIs, but provide additional ideas like *augmentation* –providing extra information– of real world elements to MAS agents. Our notion of *situatedness* could be used to provide the same normative environment to different existing systems –updating the *Bridge*–, like in [19] where they consider using the same *Normative Structure* in different *Contexts*. However, they study the adaptation of such normative context depending on individual goals, while we perform it according to institutional goals. These institutional goals could be initially agreed by participants as suggested in [20], so they will have a connection with the individual goals.

Adaptation has been usually envisioned as an agent capability where agents learn how to reorganise themselves. Thus, most works explore agent adaptation driven by individual goals. For instance, Sen an Airiau [21] study the emergence of social norms via learning from interaction experiences. The closest approach to our proposal can be found in [22] and [23]. Their agents can decide its commitment to obey norms in order to achieve associated institutional goals. In contrast, in our *Autonomic Electronic Institution* (AEI) is the organisation the one adapting itself. There are also works that include reputations models, like [24] where they collect information about norm violation/fulfilment. This information is provided to other participants, so they can adapt their social relations. Similarly, our proposal offers a third party institution that could provide this information. Currently it collects norm violations and adapts itself at system level but it could also collect and use norm fulfilment data and even provide it to participants.

6 Conclusions and Future Work

In this paper, we focus on defining adaptation and situatedness for Electronic Institutions (EI). This brings two separated agent properties to an organisational level, or, in other words, we bring up individual level capacities to global –or system– capabilities. As we have seen, we can extend EIs separately: *Autonomic Electronic Institutions* –described in section 4– include adaptation whereas *Situated Electronic Institutions* –in section 3– incorporate situatedness. Nevertheless, since both capacities are compatible, it is also possible to extend EIs with both of them simultaneously. This yields to the concept of *Situated Autonomic Electronic Institution* (SAEI) which incorporates all previously defined elements[6].

We described –in section 3– how an EI can be situated over an existing social system to try to regulate it with previously defined conventions. Moreover, we

[6] $SAEI = \langle DF', DC, B, G, TF \rangle$, $DF'\&B$ are described in section 3, DC is explained in section 2 and $G\&TF$ are defined in section 4.

also suggested –in section 4– how its adaptation capacity may be used to update such original conventions depending on institutional goals. Thus, a SAEI can be used as a tool to analyse an existing social system behaviour, and autonomously decide to modify its agent coordination to enhance its performance upon certain defined goals. Accordingly, we see a SAEI as an adaptive *coordination support* layer for social systems with two instruments: (1) supervision of social conventions and (2) adaptation of these conventions, both to enhance agent coordination, and thus, overall performance.

As future work, we envision the institution having another *coordination support* instrument: to provide assistance to its participants in form of *suggestions*. These *suggestions* would be indications about what participants should do during their interactions. Thus, for example, if an agent is trying to perform an action that is not currently allowed, suggestions may inform about those violated restrictions it is not taking into consideration that are preventing it to do the action. This mechanism would also contribute to improve agent *coordination* to enhance global performance. We also plan to include artifacts in the real world, like *intelligent objects* –objects with delegated control capacities [25]– to support this *suggestion* mechanism in situated institutions.

Acknowledgments. This work was partially funded by IEA (TIN2006-15662-C02-01), AT (CONSOLIDER CSD2007-0022), MAT2005-07244-C03-03, INGENIO2010 projects, by EU-FEDER funds, by the Catalan Gov. (Grant 2005-SGR-00093) and by Marc Esteva's Ramon y Cajal contract.

References

1. Lewis, D.: Convention: A Philosophical Study. Harvard University Press (1969)
2. Esteva, M.: Electronic Institutions: from specification to development. IIIA PhD Monography. vol. 19 (2003)
3. Boissier, O., Hübner, J.F., Sichman, J.S.: Organization oriented programming: From closed to open organizations. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS, vol. 4457, pp. 86–105. Springer, Heidelberg (2007)
4. Gomez-Sanz, J., Gervais, M.P., Weiß, G.: A survey of agent-oriented software engineering research. Kluwer, Dordrecht (2004)
5. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
6. Arcos, J.L., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: E4MAS through Electronic Institutions. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 184–202. Springer, Heidelberg (2007)
7. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer 36(1), 41–50 (2003)
8. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Self-adaptation in autonomic electronic institutions through case-based reasoning. In: Proceedings of MA4CS Workshop of ECCS 2007 (2007)
9. Sierra, C., Rodríguez-Aguilar, J.A., Noriega, P., Esteva, M., Arcos, J.L.: Personal communications about Electronic Institutions (2007)

10. Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.-A., Vasconcelos, W.: Distributed norm management in regulated mas. In: AAMAS 2007 (2007)
11. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: A rule-based approach to norm-oriented programming of electronic institutions. ACM SIGecom Exchanges 5(5), 33–40 (2006)
12. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Norm adaptation of autonomic electronic institutions with multiple goals. ITSSA 1(3), 227–238 (2006)
13. Boissier, O., Gâteau, B.: Normative multi-agent organizations: Modeling, support and control, draft version. In: Boella, G., van der Torre, L., Verhagen, H. (eds.) Normative Multi-agent Systems. Number 07122 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
14. Weyns, D., Holvoet, T.: A formal model for situated multi-agent systems. Fundam. Inform. 63(2-3), 125–158 (2004)
15. Ferber, J., Muller, J.P.: Influences and reaction: A model of situated multiagent systems. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi–Agent Systems. MIT Press, Cambridge (1995)
16. Ricci, A., Viroli, M., Omicini, A.: CArtA gO: A framework for prototyping artifact-based environments in MAS. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS, vol. 4389, pp. 67–86. Springer, Heidelberg (2007)
17. Saunier, J., Balbo, F., Badeig, F.: Environment as active support of interaction. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS, vol. 4389, pp. 87–105. Springer, Heidelberg (2007)
18. Valckenaers, P., Sauter, J.A., Sierra, C., Rodríguez-Aguilar, J.A.: Applications and environments for multi-agent systems. Autonomous Agents and Multi-Agent Systems 14(1), 61–85 (2007)
19. Cardoso, H.L., Oliveira, E., Pesquisa, N.: A Context-based Institutional Normative Environment. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 140–155. Springer, Heidelberg (2008)
20. Gaertner, D., Toni, F., Aguilar, J.A.R.: Agreeing on institutional goals for multi-agent societies. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 1–16. Springer, Heidelberg (2008)
21. Sen, S., Airiau, S.: Emergence of norms through social learning. In: Veloso, M.M. (ed.) IJCAI, pp. 1507–1512 (2007)
22. López-López, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: AAMAS, pp. 674–681. ACM, New York (2002)
23. Vecht, B., Dignum, F., Meyer, J.-J., Dignum, V.: Organizations and autonomous agents: Bottom-up dynamics of coordination mechanisms. In: Hubner, J.F. (ed.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 17–32. Springer, Heidelberg (2009)
24. Silva, V., Traba, R.H., Centeno, R.: A hybrid reputation model based on the use of organizations. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428. Springer, Heidelberg (2008)
25. Rodriguez, I., Salamo, M., Lopez-Sanchez, M., Cerquides, J., Puig, A., Sierra, C.: Completing the virtual analogy of real institutions via iobjects. In: Congrés Internacional de l'Associació Catalana d'Intelligéncia Artificial (2007)

A Context-Based Institutional Normative Environment

Henrique Lopes Cardoso and Eugénio Oliveira

LIACC, DEI / Faculdade de Engenharia, Universidade do Porto
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
{hlc,eco}@fe.up.pt

Abstract. We explore the concept of an agent-based Electronic Institution including a normative environment that supports electronic contract formation by providing a contextual normative background. We formalize the normative state using first-order logic and define institutional rules and norms operating on that state. A suitable semantics regarding the use of norms within a hierarchical context structure is given, based on norm activation conflict and defeasibility. Norm activation relies on substitution as in first-order logic. Reasoning about the fulfillment and violation of deadline obligations is formalized using linear temporal logic; implementation with institutional rules is discussed. Examples exploiting the normative environment are given.

Keywords: Normative Environment, Context, Norm Activation, Defeasibility.

1 Introduction

Electronic Institutions [1][2][3] have been proposed and developed as frameworks embedding normative environments for open multi-agent systems, where heterogeneous and independently developed agents interact. Differences exist concerning the conceptual views of the "institutional environment". In [1] a restrictive "rules of the game" approach is followed, where the institution fixes what agents are allowed to do; norms are in this case a set of interaction conventions that agents must conform to. In [2] the institution is seen as an external entity that ascribes institutional powers and normative positions, while admitting norm violations and prescribing appropriate sanctions.

In our perspective [3], an *Electronic Institution (EI)* is a software framework embracing a set of *services* and a *normative environment*. Those services are meant to assist software agents in the process of creating organizational structures ruled by a set of mutual commitments, which in the end are translated into norms. Such norms are part of the normative environment that is maintained by the *EI*. In fact, one of the core services that we consider is the provision of a supportive normative framework in the institutional environment, which agents can exploit in order to establish their contracts in a more straightforward fashion. Contracts [4] can be underspecified, relying on a structured normative framework that fills in any omissions.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 140–155, 2009.

The purpose of this paper is to formalize this normative environment. We define the notion of normative context, based on which a hierarchical structure provides a normative background for electronic contracts. Within that structure, we characterize the normative state of the system and define rules and norms operating on that state. We give a proper semantics for norms in our system by defining norm activation conflict and by providing an approach for conflict resolution based on defeasibility. We also detail the semantics of deontic statements (namely obligations with deadlines) using temporal logic, and discuss implementation issues.

The paper is organized as follows. Section 2 presents our institutional normative environment, based on context structures, including the normative state, rules and norms. Section 3 describes the semantics associated with norms, including defeasibility; deadline obligation semantics is also explored and implemented with rules. In Section 4 we illustrate the exploitation of the normative environment. Section 5 concludes and discusses related work.

2 An Institutional Normative Environment

We explore the concept of an agent-based EI including a normative environment as its core component. In the following definitions we try to provide a sound presentation of concepts in order to explain the use of norms within the normative environment.

Definition 1. *Normative Environment* $NE = \langle NS, IR, N \rangle$
The normative environment NE of an EI is composed of a normative state NS, a set IR of institutional rules that manipulate that normative state and a set N of norms, which can be seen as a special kind of rules.

While norms (see Def. 8) define the normative positions of each agent, the main purpose of institutional rules (see Def. 7) is to relate the normative state with the standing normative positions. A typical use of institutional rules is illustrated in subsection 3.2, where they are employed to implement the semantics of deadline obligations – rules monitor the normative state *NS* in order to detect the fulfillment or violation of deontic statements. On the other hand, norms "produce" those deontic statements upon certain normative state conditions.

2.1 Contexts

Our model is based on a contextualization of both the normative state and norms. In this subsection we properly introduce the notion of context and context organization.

Definition 2. *Context* $C = \langle PC, CA, CI, CN \rangle$
A context C is an organizational structure within which a set CA of agents commits to a joint activity partially regulated by a set $CN \subseteq N$ *of appropriate norms. A context includes a set CI of contextual info that makes up a kind of*

background knowledge for that context (see Def. 4). PC is the parent context within which context C is formed. Let PCA be the set of agents in context PC: we have that $CA \subseteq PCA$.

Contexts allow us to organize norms according to a hierarchical normative structure. Norm set N is partitioned into the several contexts that may exist, that is, sets CN for each context are mutually disjoint. Typically, we will have $CN \subset N$, in which case more than one context has a non-empty set CN; only if all norms in N are defined in the same context we may have $CN = N$. A norm inheritance mechanism, as explained later, justifies the fact that the locally-defined set CN of norms only *partially* regulates the activity of agents in set CA. We identify a *top* level context from which all other contexts are (directly or indirectly) formed; every agent is committed to the top context.

We now introduce the notion of *sub-context*.

Definition 3. *Sub-context* $C' = \langle PC', CA', CI', CN' \rangle$
A context C' *is a sub-context of a context* $C = \langle PC, CA, CI, CN \rangle$, *denoted* $C' \lhd C$, *if* $PC' = C$ *or if* $PC' \lhd C$. *When* C' *is either a sub-context of* C *or* C *itself, we write* $C' \trianglelefteq C$. *From Def. 2 we also have that* $CA' \subseteq CA$.

A sub-context defines a sub-activity committed to by a subset of the original context's agents. Notice that the sub-context relationship is an explicit one. Every context is a sub-context of the *top* context.

We now turn to the definition of background information that may be defined as a foundational element of a context.

Definition 4. *Contextual info* $Info^C$
Contextual info $Info^C$ *is a fully-grounded atomic formula in first-order logic, which comprises founding information regarding a context* $C = \langle PC, CA, CI, CN \rangle$. $Info^C \in CI$.

The CI component in a context definition is therefore composed of first-order logic formulae that provide background information for that context.

A B2B analogy to this kind of context/sub-context relationship comes from the virtual organizations realm, wherein a group of enterprises seeks to build a mutually beneficial relationship regarding a specific business domain. They would form a contractual agreement within the top institutional context. Often, a contract is dependent on the existence of another business relation, which forms the business context for the new contract. Each contract must contain a set of definitions regarding the role of the participants, the values to be exchanged (products or services) and their provision. In our model, these comprise information that is intrinsic and foundational to the context associated with this contract – hence the term *contextual info*.

In Section 4 a supply-agreement contract is described, in which a set of agents agrees to supply certain resources under certain conditions. In that context, contextual info is expressed as first-order formula relating each agent with a resource it supplies, together with an associated price: $supply\text{–}info^C(Ag, Res, UPr)$.

2.2 Normative State

The normative state is organized through contexts. The normative state concerns the description of what is taken for granted in a model of so-called institutional reality [5]. Therefore, we call every formula in *NS* an institutional reality element, or *IRE*. Each *IRE* refers to a specific context within which it is relevant. There can be more than one *IRE* pertaining to the same context.

Definition 5. *Contextual institutional reality element* IRE^C
A contextual institutional reality element IRE^C *is an IRE regarding context* C.
We distinguish the following kinds of IRE^C *with the following meanings:*
$ifact^C(f,t)$ – *institutional fact* f *has occurred at time* t
$time^C(t)$ – *instant* t *has elapsed*
$obl^C(a,f,d)$ – *agent* a *is obliged to bring about fact* f *until deadline* d
$fulf^C(a,f,t)$ – a *has fulfilled, at time* t*, his obligation to bring about* f
$viol^C(a,f,t)$ – a *has violated, at time* t*, his obligation to bring about* f

Note that the use of context C as a superscript is only a syntactical convenience – both contextual info and institutional reality elements are first-order formulae (C could be used as the first argument of each of these formulae). While contextual info is confined to background information that is part of the context definition, contextual institutional reality elements represent occurrences taking place after the context's creation, during its lifetime.

We consider institutional facts as *agent-originated*, since they are obtained as a consequence of some agent action [4]. The remaining elements are *environment events*, asserted in the process of norm application and monitoring. Our model of institutional reality is based on a discrete model of time. The *time* elements are used to signal instants that are relevant to the context at hand. Obligations are deontic statements, and we admit both their fulfillment and violation.

Definition 6. *Normative State* $NS = \{IRE_1^{C1}, IRE_2^{C2}, ..., IRE_n^{Cm}\}$
The normative state NS is a set of fully-grounded atomic formulae IRE_i^{Cj}, $1 \leq i \leq n$, *in first-order logic.*

The normative state will contain, at each moment, all elements that characterize the current state of affairs in every context. In that sense, *NS* could be seen as being partitioned among the several contexts, as is the case with norms; however, *IRE*'s are not part of a context's definition, since they are obtained at a later stage, during the context's operation. Some of the *IRE*'s are interrelated: for instance, a fulfillment connects an obligation to bring about a fact with its achievement as an institutional fact. These interrelations are captured with institutional rules.

2.3 Rules and Norms

Given the "contextualization" of the normative state, we are now able to define rules and norms. Institutional rules allow us to maintain the normative state of the system. They are not contextualized, but yet they operate on contextual *IRE*'s.

Definition 7. *Institutional rule* $R ::= Antecedent \rightarrow Consequent$
An institutional rule R defines, for a given set of conditions, what other elements should be added to the normative state. The rule's Antecedent is a conjunction of patterns of IRE^C *(see Def. 5), which may contain variables; restrictions may be imposed on such variables through relational conditions. We also allow the use of negation (as failure):*

$$Antecedent ::= IRE^C \mid Antecedent \wedge Antecedent \mid \neg Antecedent \mid RelCondition$$

The rule's Consequent is a conjunction of IRE^C *which are not deontic statements (*$IRE\text{–}^C$*), and which are allowed to contain bounded variables:*
$$Consequent ::= IRE\text{–}^C \mid Consequent \wedge Consequent$$

When the antecedent matches the normative state using a first-order logic substitution Θ, and if all the relational conditions over variables hold, the atomic formulae obtained by applying Θ to the consequent of the rule are added to the normative state as fully-grounded elements.

Besides institutional reality elements, the norms themselves are also contextual.

Definition 8. *Norm* $N^C ::= Situation \rightarrow Prescription$
A norm N^C *is a rule with a deontic consequent, defined in a specific context C. The norm is applicable to a context* $C' \trianglelefteq C$*. The norm's Situation is a conjunction of patterns of* $Info^{C'}$ *and* $IRE\text{–}^{C'}$ *(no deontic statements). Both kinds of patterns are allowed to contain variables; restrictions may be imposed on such variables through relational conditions:*

$$Situation ::= Info^{C'} \mid IRE\text{–}^{C'} \mid Situation \wedge Situation \mid RelCondition$$

The norm's Prescription is a (possibly empty) conjunction of deontic statements (obligations) which are allowed to contain bounded variables and are affected to the same context C'*:*

$$Prescription ::= \epsilon \mid OblConj$$
$$OblConj ::= obl^{C'}(...) \wedge OblConj \mid obl^{C'}(...)$$

Conceptually, the norm's *Situation* can be seen as being based on two sets of elements: *background (Sb)* and *contingent (Sc)*. Background elements are those that exist at context creation (the founding contextual info), while contingent elements are those that are added to the normative state at a later stage. This distinction will be helpful when describing norm semantics.

Observe the distinction between the context where the norm is defined, and the context to which the norm applies. While, in order to make the model as simple as we can, we define a norm as being applicable to a specific context, in Section 3.1 we relax this assumption, which will in part clarify the usefulness of the model.

3 Semantics

After defining each component of our normative environment, we now proceed to defining the semantics of norms and deontic statements.

3.1 Norms and Contexts

We now turn our attention to norm applicability according to the normative state. For that, we make use of the notion of *substitution* in first-order logic. We denote by $f \cdot \Theta$ the result of applying substitution Θ to atomic formula f.

Definition 9. *Norm activation*
A norm $N^C = S \rightarrow P$, applicable to a context $C' = \langle PC', CA', CI', CN' \rangle$, is said to be activated if there is a substitution Θ such that:

- $\forall_{c \in Sc}\ c \cdot \Theta \in NS$, *where Sc is the set of contingent conjuncts (IRE$^{-C'}$ patterns) in S; and*
- $\forall_{b \in Sb}\ b \cdot \Theta \in CI'$, *where Sb is the set of background conjuncts (Info$^{C'}$ patterns) in S; and*
- *all the relational conditions over variables hold.*

We are now able to define the notion of conflicting norm activations, as follows.

Definition 10. *Norm activation conflict*
Let Act_1 be the activation of norm $N_1^{C1} = S_1 \rightarrow P_1$ obtained with substitution Θ_1 and Act_2 the activation of norm $N_2^{C2} = S_2 \rightarrow P_2$ obtained with substitution Θ_2. Let $NS_1 = \{c \cdot \Theta_1 | c \in Sc_1\}$, and $NS_2 = \{c \cdot \Theta_2 | c \in Sc_2\}$, where Sc_1 and Sc_2 are the sets of contingent conjuncts of S_1 and S_2, respectively. Both NS_1 and NS_2 represent fractions of the whole normative state NS. Norm activations Act_1 and Act_2 are in conflict, written $Act_1 \bigotimes Act_2$, if $NS_1 = NS_2$ and either $C1 \lhd C2$ or $C2 \lhd C1$.

Succinctly, we say there is a norm activation conflict if we have two applicable norms activated with the same fraction of the normative state and defined in different contexts. Notice that the fact that both norms are activated with the same contextual *IRE*'s already dictates that the norm contexts, if different, have a sub-context relationship (there is no multiple inheritance mechanism in our normative structure). This becomes clearer when taking into account the sub-context (Def. 3) and norm (Def. 8) definitions: a context has a single parent context, and a norm N^C applies to a context $C' \trianglelefteq C$.

In principle, all norm activations are defeasible, according to the following definition.

Definition 11. *Norm activation defeasance*
A norm activation Act_1 for norm N_1^{C1} defeats a norm activation Act_2 for norm N_2^{C2} if $Act_1 \bigotimes Act_2$ and $C1 \lhd C2$.

A defeated norm activation is discarded, that is, the defeated activation is not applied to the normative state fraction used for activating the norm. Only undefeated norm activations will be applied: the substitution that activated a norm is applied to its prescription part and the resulting fully-grounded deontic statements are added to the normative state (recall that there are no free variables in the prescription part of norms). Observe that we do not talk about norm defeasance, but rather norm activation defeasance. Thus, the defeasance relationship may only materialize on actual norm applicability.

Norm Contextual Target. A question that may arise when going through the previous definitions can jeopardize the purpose of having defeasible norms as those in the model presented. Why should there be norms that, while being applicable to the same context, are defined in different contexts that have a sub-context relationship? Why not have all norms applicable to context C defined inside context C?

The reason for our approach becomes apparent when considering the stated aim of a supportive normative environment: to have a normative background that can fill-in details of sub-contexts that are created later and that can benefit from this setup by being underspecified. This leads us to the subject of "default rules" in the law field [6]. Thus, part of the normative environment's norms will typically be predefined, in the sense that they are pre-existent to the applicable contexts themselves. What we need is to typify contexts in order to be able to say that a norm applies to a certain type of contexts. This way, a norm might be defined at a super-context and applicable to *a range of* sub-contexts (of a certain type) to be subsequently created.

We can do this adaptation by considering context identifier C as a pair *id*:*type*, where *id* is a context identifier and *type* is a predefined context type. In a norm $N^C = S \rightarrow P$ (see Def. 8), patterns of $Info^{C'}$ and $IRE^{C'}$ inside S, as well as obligations inside P, will be rewritten to accommodate this kind of context reference, eventually using a variable in place of the context *id*. For instance, an $IRE^{Id:x}$ pattern, where *Id* is a variable, would match *IRE*'s of any sub-context of type x. When activating a norm with this kind of pattern, the substitution Θ (as used in Def. 9) would have to bind *Id* to a specific sub-context identifier; every further occurrence of *Id* is thus a bounded-variable.

This approach allows us to maintain our definitions of norm activation conflict and defeasance, with minor syntactical changes.

3.2 Deadline Obligations

Our definition of norm includes the set of conditions upon which one or more deontic statements come into being. As such, obligations being added to the normative state are no longer conditional: they are deadline obligations, in the sense discussed in [7].

In the following explanation we borrow some operators from linear temporal logic (LTL) [8]. In LTL time is assumed to be discrete, has an initial moment with no predecessors, and is infinite into the future. Let $x = (s_0, s_1, s_2, ...)$ be a timeline, defined as a sequence of states s_i. The syntax $x \models p$ reads that p is true in timeline x. We write x^k to denote state s_k of x, and $x^k \models p$ to mean that p is true at state x^k.

The following operators shall be used:

- until (U): $x \models (p\ U\ q)$ *iff* $\exists_j(x^j \models q$ *and* $\forall_{k<j}(x^k \models p))$
- before (B): $x \models (p\ B\ q)$ *iff* $\forall_j(x^j \models q$ *implies* $\exists_{k<j}(x^k \models p))$
- henceforth (G): $x \models Gq$ *iff* $\forall_j(x^j \models q)$

The *fulf* and *viol* terms in Def. 5 allow us to reason about the fulfillment and violation of obligations. Using these terms, a deadline obligation $obl^C(a, f, t)$ has the following semantics in LTL[1]:

$$\begin{gathered}(\neg ifact^C(f, \_) \wedge \neg time^C(t) \wedge \neg fulf^C(a, f, \_) \wedge \neg viol^C(a, f, \_)) \\ U \\ (ifact^C(f, t') \wedge \neg time^C(t) \wedge Gfulf^C(a, f, t') \wedge G\neg viol^C(a, f, \_)) \vee \\ (\neg ifact^C(f, \_) \wedge time^C(t) \wedge G\neg fulf^C(a, f, \_) \wedge Gviol^C(a, f, t)) \quad (1)\end{gathered}$$

This means that no violations can occur before the deadline, nor fulfillments before accomplishments; also, fulfillments and violations are mutually exclusive and persist over time.

In order to make the above formalization more tractable, we relate a deadline obligation with conditions for its fulfillment and violation. In LTL we express these relationships with:

$$obl^C(a, f, t) \wedge (ifact^C(f, t') \ B \ time^C(t)) \Rightarrow Gfulf^C(a, f, t') \quad (2)$$

$$obl^C(a, f, t) \wedge (time^C(t) \ B \ ifact^C(f, \_)) \Rightarrow Gviol^C(a, f, t) \quad (3)$$

With this approach, we are basically depending on which comes first: the deadline or the accomplishment of the fact. But in a model of discrete time, they can occur simultaneously (which is captured by operator @ defined below). In this case none of the above implications apply, therefore we add:

$$obl^C(a, f, t) \wedge (ifact^C(f, t) \ @ \ time^C(t)) \Rightarrow Gfulf^C(a, f, t) \quad (4)$$

where[2] $(\rho \ @ \ \delta) \equiv (\neg\rho \ U \ \delta) \wedge (\neg\delta \ U \ \rho) \equiv \neg(\rho \ B \ \delta) \wedge \neg(\delta \ B \ \rho)$.

We want obligations not to persist after the deadline. This allows us to model, within this framework, both cases of legal obligations, namely obligations that stand even when violated and those that do not. For instance [7], an obligation to pay for a fine will persist if it is not fulfilled until the deadline, while an obligation to submit a conference paper will not persist after the submission deadline (because submitting makes no sense at that stage). For modeling a standing obligation, the obligation can be reinstated after a violation is detected.

This property can be stated in a more general way: a fulfilled obligation cannot be violated anymore, and a violated obligation cannot be fulfilled anymore.

$$obl^C(a, f, t) \wedge fulf^C(a, f, \_) \Rightarrow G\neg viol^C(a, f, \_) \quad (5)$$

$$obl^C(a, f, t) \wedge viol^C(a, f, t) \Rightarrow G\neg fulf^C(a, f, \_) \quad (6)$$

These relationships weaken the obligation's power after it has been fulfilled or violated.

[1] The time arguments in *ifact*, *fulf* and *viol* are omitted except when they have a correspondence, as expressed in (2) and (3).

[2] $(\rho \ @ \ \delta)$ could also be defined as $x \models (\rho \ @ \ \delta)$ *iff* $\exists_j(x^j \models (\rho \wedge \delta)$ *and* $\forall_{k<j}(x^k \models (\neg\rho \wedge \neg\delta)))$.

Implementation with Institutional Rules. As mentioned before, the normative environment (Def. 1) includes a set *IR* of institutional rules (Def. 7) that manipulate the normative state. Such rules allow us to implement the semantics of deontic statements, as defined above. The *fulf* and *viol* terms in Def. 5 are meant to allow us to reason about the fulfillment and violation of obligations as soon as they occur, by defining norms that take these elements into account in their antecedent. Institutional rules enable the specification of conditions for fulfillment and violation detection.

According to the deadline obligation semantics described above, namely (2) and (3), we may have the following institutional rules (where variables begin with an upper-case letter):

$$obl^C(A, F, T) \wedge ifact^C(F, T') \wedge \neg time^C(T) \rightarrow fulf^C(A, F, T') \tag{7}$$

$$obl^C(A, F, T) \wedge time^C(T) \wedge \neg ifact^C(F, \_) \rightarrow viol^C(A, F, T) \tag{8}$$

But what if both the fact and the deadline hold at some point in time? If $(ifact^C(f, \_)\ B\ time^C(t))$, then rule (7) asserted a fulfillment; on the other hand, if $(time^C(t)\ B\ ifact^C(f, \_))$ then rule (8) asserted a violation. But what if $(ifact^C(f, \_)\ @\ time^C(t))$? A rule like:

$$obl^C(A, F, T) \wedge ifact^C(F, T') \wedge time^C(T) \rightarrow fulf^C(A, F, T') \tag{9}$$

is not acceptable, as it would apply if $(time^C(t)\ B\ ifact^C(f, \_))$. We need to keep the property that after being violated, the obligation cannot be fulfilled anymore (as in (6) above). We may say:

$$obl^C(A, F, T) \wedge ifact^C(F, T') \wedge \neg viol^C(A, F, \_) \rightarrow fulf^C(A, F, T') \tag{10}$$

It is tempting to also explicitly state that violations can only occur if no fulfillment was achieved before. Something like:

$$obl^C(A, F, T) \wedge time^C(T) \wedge \neg fulf^C(A, F, \_) \rightarrow viol^C(A, F, T) \tag{11}$$

However, when taken together with (10), this would imply that a simultaneous occurrence of $ifact^C(f, \_)$ and $time^C(t)$ (that is, $ifact^C(f, \_)\ @\ time^C(t)$) could bring either a fulfillment or a violation! We therefore must join (8) with (10). (Notice that the pairing of (7) with (11) would bring a violation in the simultaneity case.)

Practical Issues. If we cannot assume that the above rules are evaluated at every normative state update, we may get unwanted results. For instance, assume that the following are elements of the current normative state: $obl^C(a, f, t)$, $time^C(t)$ and $ifact^C(f, t + 1)$. If rules are applied only at time $t' > t$, the violation would go unnoticed: rule (10) would apply, while rule (8) would not.

This problem can be overcome by referring explicitly to the time references of *IRE*'s:

$$obl^C(A, F, T) \wedge time^C(T) \wedge \neg(ifact^C(F, T') \wedge T' \leq T) \rightarrow viol^C(A, F, T) \tag{12}$$

$$obl^C(A, F, T) \wedge ifact^C(F, T') \wedge T' \leq T \rightarrow fulf^C(A, F, T') \tag{13}$$

If we are to relax the rule evaluation policy, the two rules for fulfillment and violation detection must become independent. The shortcoming of this approach is that it is directly applicable only when considering temporal deadlines.

Another problem that we do not consider is that we are assuming an instant recognition of each *IRE*. That is, an institutional fact occurring at time t is added at that same instant t to the normative state. Were that not the case, we could get into situations where certain violations would need to be retracted as new knowledge is acquired, otherwise inconsistencies might be obtained (which could be avoided with an extra $\neg viol^C(A, F, \_)$ test in rule (13) above).

4 Examples

In this section we sketch some examples towards the exploitation of the normative environment. The examples are necessarily simple, in order to focus on the important aspects of our approach; in the following we adopt the convention that variables begin with an upper-case letter.

Suppose that a group of companies provide household appliance solutions to their customers. However, while these solutions involve several kinds of equipment, each of the companies manufactures only a subset of them. They agree to form a virtual organization in order to better serve their customers.

This organization will define a supply-agreement that translates into a context *sa3*:*sa* in the normative environment, where *sa3* is the context id and *sa* is the context type (see end of Section 3.1). Notice that *sa3*:*sa* $\lhd$ *top*, where *top* is the top context.

Suppose we have, at the top context, the following norm:

$$\begin{aligned} N_1^{top} = {} & ifact^{X:sa}(order(A1, Res, Qt, A2), T) \wedge \\ & supply\text{–}info^{X:sa}(A2, Res, Upr) \\ & \rightarrow \\ & obl^{X:sa}(A2, delivery(A2, Res, Qt, A1), T + 2) \wedge \\ & obl^{X:sa}(A1, payment(A1, Qt * Upr, A2), T + 2) \end{aligned}$$

The norm states that for any supply-agreement, when an order is made that corresponds to the supply information of the receiver, he is obliged to deliver the requested goods and the sender is obliged to make the associated payment.

Table 1. Different normative states and norm activation conflicts

NS	$\{ifact^{sa3:sa}(order(jim, r3, 5, tom), 1)\}$
Conflict	none, N_1^{top} applies
NS'	$\{ifact^{sa3:sa}(order(jim, r3, 5, tom), 1), obl^{sa3:sa}(tom, delivery(tom, r3, 5, jim), 3), obl^{sa3:sa}(jim, payment(jim, 5, tom), 3)\}$
NS	$\{ifact^{sa3:sa}(order(tom, r1, 5, jim), 1)\}$
Conflict	none, N_1^{top} applies
NS'	$\{ifact^{sa3:sa}(order(tom, r1, 5, jim), 1), obl^{sa3:sa}(jim, delivery(jim, r1, 5, tom), 3), obl^{sa3:sa}(tom, payment(tom, 5, jim), 3)\}$
NS	$\{ifact^{sa3:sa}(order(tom, r1, 100, jim), 1)\}$
Conflict	$N_1^{sa3:sa}$ defeats N_1^{top}
NS'	$\{ifact^{sa3:sa}(order(tom, r1, 100, jim), 1), obl^{sa3:sa}(jim, delivery(jim, r1, 100, tom), 6), obl^{sa3:sa}(tom, payment(tom, 100, jim), 3)\}$
NS	$\{ifact^{sa3:sa}(order(sam, r3, 5, tom), 1)\}$
Conflict	$N_2^{sa3:sa}$ defeats N_1^{top}
NS'	$\{ifact^{sa3:sa}(order(sam, r3, 5, tom), 1), obl^{sa3:sa}(tom, delivery(tom, r3, 5, sam), 3)\}$
NS	$\{ifact^{sa3:sa}(order(sam, r3, 5, tom), 1), obl^{sa3:sa}(tom, delivery(tom, r3, 5, sam), 3), fulf^{sa3:sa}(tom, delivery(tom, r3, 5, sam), 2)\}$
Conflict	none, $N_3^{sa3:sa}$ applies
NS'	$\{ifact^{sa3:sa}(order(sam, r3, 5, tom), 1), obl^{sa3:sa}(tom, delivery(tom, r3, 5, sam), 3), fulf^{sa3:sa}(tom, delivery(tom, r3, 5, sam), 2), obl^{sa3:sa}(sam, payment(sam, 5, tom), 4)\}$

Now, suppose context *sa3*:*sa* includes the following norms.

$$\begin{aligned} N_1^{sa3:sa} = {} & ifact^{sa3:sa}(order(A1, Res, Qt, jim), T) \wedge \\ & supply\text{–}info^{sa3:sa}(jim, Res, Upr) \wedge Qt > 99 \\ & \rightarrow \\ & obl^{sa3:sa}(jim, delivery(jim, Res, Qt, A1), T + 5) \wedge \\ & obl^{sa3:sa}(A1, payment(A1, Qt * Upr, jim), T + 2) \end{aligned}$$

This norm expresses the fact that agent *jim*, when receiving orders with more than 99 units, has an extended delivery deadline.

$$\begin{aligned} N_2^{sa3:sa} = {} & ifact^{sa3:sa}(order(sam, Res, Qt, A2), T) \wedge \\ & supply\text{–}info^{sa3:sa}(A2, Res, \_) \\ & \rightarrow \\ & obl^{sa3:sa}(A2, delivery(A2, Res, Qt, sam), T + 2) \end{aligned}$$

$$\begin{aligned} N_3^{sa3:sa} = {} & fulf^{sa3:sa}(A2, delivery(A2, Res, Qt, sam), T) \wedge \\ & supply\text{–}info^{sa3:sa}(A2, Res, Upr) \\ & \rightarrow \\ & obl^{sa3:sa}(sam, payment(sam, Qt * Upr, A2), T + 2) \end{aligned}$$

These two norms express the higher position of agent *sam* who, as opposed to other agents, only pays after receiving the merchandise. Suppose we have the following founding contextual info for context *sa3*:*sa*:

$$supply\text{–}info^{sa3:sa}(jim, r1, 1)$$
$$supply\text{–}info^{sa3:sa}(sam, r2, 1)$$
$$supply\text{–}info^{sa3:sa}(tom, r3, 1)$$

Table 1 shows what might happen in different normative states. Lines labeled with *Conflict* in the first column show what norm activation conflicts come about (and how they are resolved) when the institutional reality elements of their previous line (labeled with NS) are present. Lines labeled with NS' show the normative state after applying the defeating norm activation. Notice that in the second example there is no conflict, since norm $N_1^{sa3:sa}$ is not activated because of a variable restriction.

Observe that the model is very flexible, allowing us to specify different contracting situations where the concept of norm activation defeasibility is useful.

5 Conclusions and Related Work

Our model of Electronic Institution [3][4] is based on an environment with a hierarchical normative structure, including norm inheritance as a mechanism to facilitate contract establishment. This paper formalizes such an environment.

We rely on a common normative structure applicable to several "social systems", where the institution is prior in existence to the specific social relationships (which are mapped to contexts). A different perspective is taken in [9], where an electronic institution is coupled (situated) with a previously existing social system. The authors also explore the possibility of autonomic adaptation of the institution's rules to enhance performance. In our case, adaptability is addressed by having a normative environment that agents can exploit and adapt to fit their purposes.

The idea of *context* for normative reasoning has been studied before. However, in most cases the notion of context comes from the 'counts-as' relation [10][11]: "X counts-as Y in context C". For instance, in [12][13] a context gives an interpretation to abstract norms of a broader context. There is a leveled structuring of contexts, which broadly contemplates institutions, sub-institutions and organizations, from the most abstract to the most concrete level. However, concrete norms (refined as rules and implemented as procedures) are used to model preexistent organizations. Concept abstraction is studied in [14]. In this case, it is not the norm that is abstract, but instead the concepts in which it is expressed. A norm based on abstract concepts may be further specified in a more specific context. Our approach has a different concern: we use the context structure for designing a model of defeasibility for norms, which may be added to the system at runtime. We do not tackle with abstraction.

The "contextualization" of contracts within higher normative structures has also been advanced in [15]. In this case, a contract is modeled as an institution itself (see also [16]), and can be governed by another (super) institution. This relationship is expressed through a mechanism of empowerment. States are

described by fluents and evolve according to rules expecting events. Empowerments are defined by normative fluents allowing the creation of events and the initialization or termination of fluents. With this approach, a rule defined in an institution may operate on another institution's state if the rule's effects are explicitly empowered. In our approach, contracts are modeled as contexts within a single institution. Norms can also operate in contexts other than the one where they are defined, but this property is based on a structured normative framework, and not on a discretionary basis that may be cumbersome to express.

From the law field, three normative conflict resolution principles have been defined and traditionally used. The *lex superior* is a hierarchical criterion and indicates that a norm issued by a more important legal entity prevails, when in conflict with another norm (e.g. the Constitution prevails over any other legal body). The *lex posterior* is a chronological criterion indicating that the most recent norm prevails. The *lex specialis* is a specificity criterion establishing that the most specific norm prevails. While not firmly adopting any of these options, our approach resembles more the *lex specialis* principle. However, the defeating norms are more specific in the sense that they are *defined at* (as opposed to *applied to*) a more specific context (a kind of "lex inferior"). The *lex specialis* flavor comes from the fact that in most cases a defeating norm should apply to a narrower context-set.

These properties of our norm defeasance approach result from the fact that the original aim is not to impose predefined regulations on agents, but instead to help them in building contractual relationships by providing a normative background (which can be exploited in a partial way). A feature of our approach that exposes this aim is that all norms are defeasible. In this respect we follow the notion from law theory of "default rules" [6]. We leave for future work the possibility of defining non-defeasible norms, that is, norms that are not to be overridden.

This notion of "default rules" might be misleading; it has not a direct correspondence with default logic formalizations [17]. We do not handle the defeasibility of conclusions of default rules in that sense, but instead model defeasibility of the application of the rules themselves (which are called norms).

From a theoretical logical stance, norm defeasibility has been addressed in, e.g., [18][19][20]. Typically, deontic reasoning guides these approaches, and thus conflicts regard the deontic operators themselves. Our approach is centered instead on the applicability of norms, not on their conclusions.

The work in [21] addresses the issue of conflict resolution in a structured setup of compound activities. These resemble our context and sub-context relationships. However, they model deontic conflicts (e.g. an action being obliged and prohibited), while we model norm (activation) conflicts. They study the inheritance of normative positions (obligations, permissions, prohibitions), based on an explicit stamping of each one of them with a priority value and a timestamp; the specificity criterion is based on the compound activities' structure. We address the inheritance of norms and provide a means to override norm activations based on their defeasibility.

Our approach of context and sub-context definitions, together with the presented norm defeasibility model, is similar to the notion of *supererogatory defeasibility* in [22]. They model defeasibility in terms of role and sub-role definitions. In fact, they also consider *express defeasibility*, which is based on the specificity of conditions for norm applicability, but this approach has been followed by several others.

The problem of normative conflict resolution has been also studied in more practical approaches. The application of business rules in e-commerce has been addressed in [23], where courteous logic programs allow for an explicit definition of priorities among rules. An extension based on defeasible and deontic logic has been advanced in [24] for the representation of business contracts (and not merely business rules). However, this approach does not consider defeasibility of norms between a contract and an underlying normative framework. Finally, [25] also addresses defeasible reasoning in the e-contracts domain, based on the translation of contracts from event calculus to default logic, and on the definition of dynamic priorities among rules (by using domain-dependent criteria). Conflics are, in this case, based on the normative positions of agents.

We should also point out that [26] presents a grammar for rules that combines both our rule and norm definitions. However, our concern is to distinguish *a priori* rule definition as a normative state maintenance issue from norm definition as a contracting activity. Furthermore, in [26] there is no attempt to solve any disputes related with possibly conflicting norms.

From a software engineering perspective, we envisage the development of different "enterprise agents" that encapsulate the private interests of the electronic institution participants, and that engage in (partially automated) negotiations in order to obtain mutually beneficial contracts. We have a working platform that incorporates the needed infrastructure for the concepts introduced in this paper, based on Jess [27] – a very efficient rule engine based on the Rete algorithm for pattern matching. Another major effort concerns the knowledge engineering of norms applicable to different business contexts, in order to maximize the usefulness of the normative background.

Some open issues in our research include, as already mentioned, the possibility of defining non-defeasible norms, which might be important in certain contract domains. The development of multiple-inheritance mechanisms within our contextual framework is also an interesting issue, although it poses additional problems regarding norm defeasibility.

Acknowledgments. The first author is supported by FCT (Fundação para a Ciência e a Tecnologia) under grant SFRH/BD/29773/2006.

References

1. Arcos, J.L., Esteva, M., Noriega, P., Rodríuez-Aguilar, J.A., Sierra, C.: Environment engineering for multiagent systems. Engineering Applications of Artificial Intelligence 18(2), 191–204 (2005)

2. Artikis, A., Pitt, J., Sergot, M.: Animated specifications of computational societies. In: Castelfranchi, C., Johnson, W.L. (eds.) International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, pp. 1053–1062. ACM Press, New York (2002)
3. Lopes Cardoso, H., Oliveira, E.: Electronic institutions for b2b: Dynamic normative environments. Artificial Intelligence and Law 16(1), 107–128 (2008)
4. Lopes Cardoso, H., Oliveira, E.: A contract model for electronic institutions. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 27–40. Springer, Heidelberg (2008)
5. Lopes Cardoso, H., Oliveira, E.: Institutional reality and norms: Specifying and monitoring agent organizations. International Journal of Cooperative Information Systems 16(1), 67–95 (2007)
6. Craswell, R.: Contract law: General theories. In: Bouckaert, B., De Geest, G. (eds.) Encyclopedia of Law and Economics. The Regulation of Contracts, vol. III, pp. 1–24. Edward Elgar, Cheltenham (2000)
7. Dignum, F., Broersen, J., Dignum, V., Meyer, J.J.: Meeting the deadline: Why, when and how. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 30–40. Springer, Heidelberg (2004)
8. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science. Formal Models and Sematics, vol. B, pp. 995–1072. North-Holland Pub. Co./MIT Press (1990)
9. Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J.A., Esteva, M.: Formalising situatedness and adaptation in electronic institutions. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 126–139. Springer, Heidelberg (2008)
10. Searle, J.R.: The Construction of Social Reality. Free Press, New York (1995)
11. Jones, A., Sergot, M.: A formal characterisation of institutionalised power. Logic Journal of the IGPL 4(3), 427–443 (1996)
12. Vázquez-Salceda, J., Dignum, F.: Modelling electronic organizations. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 584–593. Springer, Heidelberg (2003)
13. Grossi, D., Dignum, F.: From abstract to concrete norms in agent institutions. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 12–29. Springer, Heidelberg (2004)
14. Grossi, D., Dignum, F., Meyer, J.J.C.: Contextual taxonomies. In: Leite, J., Torroni, P. (eds.) CLIMA 2004. LNCS (LNAI), vol. 3487, pp. 33–51. Springer, Heidelberg (2005)
15. Cliffe, O., De Vos, M., Padget, J.: Specifying and reasoning about multiple institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 67–85. Springer, Heidelberg (2007)
16. Boella, G., van der Torre, L.: Contracts as legal institutions in organizations of autonomous agents. In: Jennings, N., Sierra, C., Sonenberg, L., Tambe, M. (eds.) Third International Joint Conference on Autonomous Agents and Multi Agent Systems, vol. 2, pp. 948–955. ACM Press, New York (2004)
17. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13(1/2), 81–132 (1980)
18. Royakkers, L., Dignum, F.: Defeasible reasoning with legal rules. In: Brown, M., Carmo, J. (eds.) Deontic Logic, Agency and Normative Systems (Workshops in Computing), pp. 174–193. Springer, Heidelberg (1996)

19. Sartor, G.: Normative conflicts in legal reasoning. Artificial Intelligence and Law 1(2-3), 209–235 (1992)
20. van der Torre, L.: Violated obligations in a defeasible deontic logic. In: Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI 1994), pp. 371–375. John Wiley and Sons, Amsterdam (1994)
21. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: An algorithm for conflict resolution in regulated compound activities. In: Seventh Annual International Workshop Engineering Societies in the Agents World (ESAW 2006) (2006)
22. Ryu, Y.U.: Relativized deontic modalities for contractual obligations in formal business communication. In: 30th Hawaii International Conference on System Sciences (HICSS), Hawaii, USA. Information Systems Track - Internet and the Digital Economy, vol. 4, p. 485 (1997)
23. Grosof, B.N.: Representing e-commerce rules via situated courteous logic programs in ruleml. Electronic Commerce Research and Applications 3(1), 2–20 (2004)
24. Governatori, G.: Representing business contracts in ruleml. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
25. Giannikis, G.K., Daskalopulu, A.: Defeasible reasoning with e-contracts. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 690–694 (2006)
26. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: Norm-oriented programming of electronic institutions: A rule-based approach. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 177–193. Springer, Heidelberg (2007)
27. Friedman-Hill, E.: Jess in Action. Manning Publications Co. (2003)

Towards a Formalisation of Electronic Contracting Environments

Nir Oren[1], Sofia Panagiotidi[2], Javier Vázquez-Salceda[2], Sanjay Modgil[1], Michael Luck[1], and Simon Miles[1]

[1] Dept. of Computer Science
King's College London
Strand, London
WC2R 2LS, United Kingdom
{nir.oren,sanjay.modgil,michael.luck,simon.miles}@kcl.ac.uk
[2] Dept Llenguatges i Sistemes Informatics
Univ. Politecnica de Catalunya
Edifici OMEGA, c/Jordi Girona 1-3
E-08034 Barcelona, Spain
{panagiotidi,jvazquez}@lsi.upc.edu

Abstract. Clauses within contracts may be thought of as norms, specifying permissions, obligations and prohibitions on contract parties. In this paper, we present a formal representation of contracts, focusing on the specification of a model of norms. With this model, a norm is associated with a status, which may change as the environment, and the status of other norms, changes. We define a normative environment, which may be used to track the status of a set of norms throughout their lifecycle, and then describe a predicates that may be used to evaluate a norm's status. Agents are able to use these predicates to reason about the status of norms, and how their actions will affect the normative environment. Finally, we show the applicability of our framework to real world domains by monitoring the execution of a contract taken from a real world scenario.

1 Introduction

With the increasing popularity of online transactions, the need for electronic contracting has become apparent. While generic contracts are applicable in many situations, and violations sufficiently rare that a human may resolve disputes, the appearance of web-services, and the need to regulate interactions between them highlights the desirability of fully automated contracting. Such fully automated contracting requires the ability to describe a contract in a machine interpretable way, ideally in a form over which inference may be performed. Additionally, techniques for automatically generating and enforcing contracts are also required, as well as protocols allowing agents to create and modify contracts.

Work dealing with some of these areas already exists; for example, [1] discusses automated negotiation in various contexts, including in contracts, while [2,3] and [4] all suggest different types of contracting languages. At their core, contracts are primarily normative documents; that is they impose a set of (possibly conditional) requirements on agent behaviour. These requirements may range from actions that the agent may,

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 156–171, 2009.

or should, undertake, to states of affairs within the environment that an agent may, should, or should not, allow to occur. To formalise a contracting language, one must thus first formalise its normative components. As discussed later, researchers have provided many such formalisations, often in the context of deontic logic. Our interest in norms is more focused; as part of a contracting language, we are interested in tracking the changing state of norms (for example, when they are *active*, that is, have normative force on an agent, as well as the more traditional *violated*). Furthermore, our application domain requires slightly different philosophical assumptions when compared to those made in the deontic tradition, as we assume that norms can be violated, but may then, in some cases, be *unviolated*. For example, consider a norm stating that whenever it is windy, Alice should ensure that the door is closed. When it is windy, the door may be open, and Alice is thus in violation of the norm. Once she (or some other agent) closes the door, the norm is unviolated. However, if the door blows open, the norm is again violated. This norm may be violated and unviolated as long as it is active, that is, as long as it is windy.

In this paper, we present a formal normative framework that allows us to track the changing status of norms. The remainder of the paper is structured as follows: we begin by providing an informal overview of our normative model. We then describe a typical case in which norms may be used within a contract. This serves as a running example throughout the remainder of the paper. The normative model is then formalised in two parts. We begin by structurally describing the various elements of our model, and then show how we may capture their dynamic behaviour. After illustrating our model via an example, we conclude by discussing related and future work.

2 Norms for Modelling Contract Clauses

We assume that a contract is made up of various descriptive elements, for example stating which ontologies may be used to explain the terms found within it. Most importantly, it specifies a set of *clauses*, each of which represents a *norm*.

Norms can be interpreted as socially derived prescriptions specifying that some set of agents (the norm's *targets*) may, or must, perform some action, or see that some state of affairs occurs. Norms can thus be understood as regulating the behaviour of agents. This is their role when encoded in contracts.

Norms are social constructs, and we argue that it is meaningless to consider norms independently of their social aspect. This is because a norm is imposed on the target by some other entity (the imposer) which must be granted, via the society, some power to impose the norm. Without this power, the norm's target is free to ignore the norm's prescriptions. With the presence of power, a penalty may be imposed on an agent violating a norm. These penalties take on the form of additional norms, giving certain agents within a society *permission* to impose penalties (or *obliging* them to do so).

In designing our normative model, we are concerned with meeting the following requirements, imposed upon us by the contracting domain in which we operate.

- The model should allow for the monitoring of norms. That is, it should allow for the determination of whether a violation took place and, if possible, who was responsible for causing the violation.

- Verification of norms should also be supported, i.e. determining whether conflicts between norms could occur, or whether a norm could never or sometimes or always be complied with.
- Agents should be able to make use of the normative model to support their own practical reasoning, i.e. deciding which action they should undertake.
- Norms should be able to cope with contrary to duty obligations as well as conditions based on the status of other norms. For example, consider the pair of norms "One is obliged to park legally", and "If one parks illegally, one is obliged to pay a fine". The second norm carries normative weight only if the first norm is violated. These types of norms commonly appear within contracts, and it is thus critical that our model is able to represent them.
- The model should be extensible, allowing different knowledge representations and reasoning mechanisms to make use of it.

No requirement is placed on detecting and resolving normative conflict. Many such techniques exist, each having a different view of what constitutes normative conflict (e.g. [5,6]), and it is intended that these techniques make use of our framework for their underlying representation of norms. Similarly, our model should not prescribe what must occur when a violation is detected. Instead, we assume that the contract would contain clauses dealing with such situations.

Norms typically have normative force only in certain situations. We therefore associate norms with an *activation condition*. Norms which are not applicable to a situation are considered *abstract*, and are *instantiated* when the norm's activation condition holds. Once a norm has been instantiated, it remains active, irrespective of its activation condition, until a specific *expiration condition* holds. When the expiration condition occurs, the norm is assumed no longer to have normative force. Finally, independent of these two conditions is the norm's *normative goal*, which is used to identify when the norm is violated (in the case of an obligation), or what the agent is actually allowed to do (in the case of a permission). Obligations and permissions are the two *norm types* on which our framework focuses. Like others, we assume that additional norm types may be constructed from these basic types (for example, a prohibition could be seen as an obligation with a negated normative goal).

Norms may be activated, met and discharged based on a number of factors including the status of other norms, the state of the environment (and the actions performed by other agents therein), and the status of contracts.

3 Example: Car Insurance Brokerage

As a running example, we make use of a simplified version of a use case scenario from the IST-CONTRACT project[1]. This scenario models the agreements between several parties in the car insurance domain. One party, the *repair company*, is responsible for

[1] The IST-CONTRACT project, funded by the European Commission, aims to develop frameworks, components and tools which make it possible to model, build, verify and monitor distributed electronic business systems on the basis of dynamically generated, cross-organisational contracts. More information can be found at `http://www.ist-contract.org`.

repairing damaged cars. The second party named in the contract, is a *car insurance company* which has damaged cars that need to be repaired. A third organisation, *Damage Secure*, acts as a broker between the *insurance company* and *repair company*, facilitating all dealings between them. The goal of contracts between the parties in this use case is to enhance the quality and efficiency of the total damage claims handling process between parties.

A typical contract in this use case involves the repair of a damaged car. Here, after the *repair company* receives a damaged car and consents to its repair, *Damage Secure* and the *repair company* officially commit to a short term contract which specifies the details of the repair procedure, including the invoice, etc. Then, the repair company repairs the car and notifies *Damage Secure* when it is complete. *Damage Secure* handles the payment agreed to in the contract, provided there is no dispute over the quality of the repair (in which case an expert is called to perform a quality assessment). Additional contracts would exist between *Damage Secure* and the *insurance company* on whose behalf the repair is being carried out, but we ignore these contracts within this paper.

The focus of interest of this example is to show how a repair contract and a set of instantiated norms over the repair of a car operate within the domain and normative environment. Such a case is useful to demonstrate how a contract and the norms attached to it can be monitored throughout its execution.

4 Formalisation

In this section, we formalise our notions of norms. We do so in a number of steps: first, we define their structure; after this is done, we show how the status of a norm may change over time. Before examining norms, we must define a number of related concepts.

4.1 Formal Preliminaries

We assume the use of a predicate based first order language $\mathcal{L}$ containing logical symbols: connectives $\{\neg, \wedge, \vee, \rightarrow\}$, quantifiers $\{\forall, \exists\}$, an infinite set of variables, and the non-logical predicate, constant and function symbols. The standard definitions for free and bound variables, as well as ground formulae are assumed. Finally, the set of well formed formulae of $\mathcal{L}$ are denoted as $wff(\mathcal{L})$. A single well-formed formula from this set is denoted *wff*.

We make use of the standard notions of substitution of variables in a wff, where $S = \langle t_1/v_1, \ldots, t_n/v_n \rangle$ is a substitution of the terms $t_1, \ldots, t_n$ for variables $v_1, \ldots, v_n$ in a wff. If no variables exist in a wff resulting from a substitution, it is said to be fully grounded, and is partially grounded otherwise.

Our model allows us to infer predicates based on the status of the environment, clauses, and norms. We assume that other predicates may exist whose truth values may be inferred from other sources such as ontologies, or an action model. Each of these sources thus generates a theory, denoted by Γ. For example, we label the theory generated by the environment as Γ_{Env}. We label the union of all theories as $\mathbf{\Gamma}$.

Formally, a *contract document* contains a set of *clauses* representing *norms* imposed on *agents*. A contract document that has been agreed to by those agents has normative force, and the agents affected by a contract document's norms are the parties to that contract. Since a contract document may be instantiated more than once with different agents playing similar roles, agents are identified within a contract using an indirection mechanism: a contract document imposes norms on a set of *roles*, and agents are associated with these roles before the contract is agreed to.

While not mentioned in this document, additional types of contract documents, such as contract proposals (which represent contract documents to be agreed upon and with no normative weight), may exist. References to contracts in the rest of this paper refer to contract documents which have been agreed upon, and thus carry normative weight.

4.2 Structural Definitions

We may now define the structure of norms and contract documents. Since these concepts act upon agents, we begin by defining these entities, as well as roles, which are names referenced to identify the agent upon which a norm acts.

Agent Names and Roles. Agents in our framework are left unspecified; we only assume that they are associated with a unique agent name. This name is used to associate them to specific norms.

Contracts associate agents with roles. For example, the car insurance brokerage contract described earlier contains two roles, that of the *broker*, and that of the *repairer*. One of the clauses obliges the repairer to repair a car, while another assigns a permission to the agent taking on the broker role, allowing it to demand a penalty from the repairer if a car is not fixed by the end of a period specified within the contract. A specific agent (e.g. Bob's car repair company) may then be associated with the *repairer* role by the contract.

A contract role may have one or more parent roles. This means that whenever an agent is assigned to a role, it is also assigned to that role's parent roles, and so assumes the clauses applying to those parents. If a role r_1 is a parent of role r_2, then r_2 may be referred to as the child role of r_1.

If we would like to specify that the role of a *car repairer* exists in a contract, and also include the fact that any *car repairer* also acts as a *repairer* (i.e. *repairer* is a parent role of *car repairer*), then $(repairer, carRepairer)$ would be contained within $RoleHierarchyDefinition$.

Definition 1. ***(Roles)*** *A role is a constant. We assume that the set of all roles is called* $Roles$*. Then a role hierarchy definition* $RoleHierarchyDefinition$ *is a binary relation of the form* $(Parent, Child)$ *where* $Parent, Child \in Roles$*.*

Norms. A contract contains a set of clauses, represented by norms. Norms may bind an agent to a certain course of action in all situations, or may only affect an agent when certain activation conditions apply. Similarly, once an agent achieves a certain state of affairs, a norm may no longer apply. Finally, norms affect only a specific set of target agents. A norm thus consists of the following components.

- A type identifier, stating whether the norm is an obligation or a permission.
- An activation condition stating when the norm must be instantiated.
- A normative goal or state (condition) used to identify when the norm is violated (in the case of obligations) or what the agent is allowed to do (in the case of permissions).
- An expiration condition used to determine when the norm no longer affects the agent.
- A target, identifying which agents the norm affects.

Norms may be activated, met, and discharged based on various factors including the environment, the status of contracts, and the status of other norms. We assume the existence of $\mathbf{\Gamma}$, a theory (or possibly a set of theories) allowing one to interpret the status of norms[2]. To represent the status of a norm, we define a normative environment theory Γ_{NEnv} below, and assume that it is part of $\mathbf{\Gamma}$.

A norm that may apply to a number of situations is, in a sense, abstract. When a situation to which it applies does arise, the norm is instantiated and exerts a normative force on the agents that are beholden to it. We may thus informally define an abstract norm as a norm that, when the conditions are right, comes into effect (i.e. is instantiated) and only then has normative force over one or more agents. As the name suggests, an instantiated norm is an instantiated abstract norm, which has normative power over a set of agents, until it is discharged.

A group of abstract norms (which, in our case, are the clauses of a contract) is gathered into an abstract norm store. Norms may be represented as a tuple of *wffs*.

Definition 2. ***(Abstract Norms and Abstract Norm Store)*** *An* Abstract Norm Store*, denoted $\mathcal{ANS}$, consists of a set of abstract norms, each of which is a tuple of the form*

$$\langle NormType, NormActivation, NormCondition, \\ NormExpiration, NormTarget\rangle$$

where:
1. $NormType \in \{$*obligation, permission*$\}$
2. for $N \in \{NormActivation, NormCondition, NormExpiration, NormTarget\}$, N *is a* wff *(denoted by* ϕ_N*)*

We may further divide $NormCondition$ into a state based maintenance condition (labelled $SMaintenanceCondition$) and an action based maintenance condition (labelled $AMaintenanceCondition$). A truth value for $NormCondition$ may be computed as the truth value of $(AMaintenanceCondition \wedge SMaintenanceCondition)$.

$NormActivation$ is some *wff* ϕ_{NA} which, when entailed by the theory, must be entailed as the fully grounded ϕ'_{NA} in order that the abstract norm can be instantiated and thus come into force. The substitution of variables S such that $\phi'_{NA} = S(\phi_{NA})$ is then applied to the other components of the abstract norm, thus specifying the instantiated norm.

[2] For example, $\mathbf{\Gamma}$ may include references to the environment (a theory Γ_{Env}), an ontology, an action model, and normative environment theory.

Instantiating Abstract Norms. We now define how abstract norms are instantiated with respect to the domain environment theory and normative environment theory.

An instantiated norm has the same overall form as an abstract norm but its activation condition is grounded, and its remaining parameters are partially grounded using the same grounding as the activation condition.

Definition 3. ***(Instantiation of Abstract Norms)***
The abstract norm

$$\langle NormType, NormActivation, NormCondition, \\ NormExpiration, NormTarget\rangle$$

instantiated by a theory $\mathbf{\Gamma}$ *made up of sub theories including one representing the domain's environment (*Γ_{Env}*) and* Normative Environment Theory *(*Γ_{NEnv}*), obtains an instantiated norm:*

$$\langle NormType, NormActivation', NormCondition', \\ NormExpiration', NormTarget'\rangle$$

where:

- $\mathbf{\Gamma} \vdash NormActivation'$*, where* $NormActivation'$ *is fully grounded such that* $NormActivation'$ *= S(*$NormActivation$*)*
- $NormCondition'$ *= S(*$NormCondition$*)*
- $NormExpiration'$ *= S(*$NormExpiration$*).*
- $NormTarget' = \{X \mid \mathbf{\Gamma} \cup \{NormActivation'\} \cup \{S(NormTarget)\} \vdash X\}$*, where* $NormTarget' \subseteq AgentNames$

Notice that $NormTarget'$ is the set of individuals X to whom the instantiated norm applies. These individuals are identified with reference to (entailed by) the domain environment theory, normative environment theory[3], and the $NormActivation$ and $NormTarget$ *wffs* that are grounded with respect to the former environments. In the context of a contract clause, the norm's targets are identified by making use of the $RoleHierarchyDefinition$ relation. Note also that both $NormCondition'$ and $NormExpiration'$ may only be partially grounded.

Given a set of abstract norms, ANS, together with a $\mathbf{\Gamma}$, we define the set of norms that may be instantiated from $\mathbf{\Gamma}$ as $inst(ANS)$.

Contracts. Given the definition of roles, we may specify a contract document as follows.

Definition 4. ***(Contract Document)*** *A Contract Document is a tuple*

$$ContractDocument = \langle \mathbf{\Gamma}, CDNorms, CDRoles, \\ CDRoleMapping\rangle$$

[3] The normative environment theory may be used when a target should be identified based on the status of another norm. For example, in the case of a contrary to duty obligation, a penalty must be paid by the agent(s) violating some other norm.

where $CDNorms$ is a set of abstract norms. $CDRoles$ is a set of role definitions, and $CDRoleMapping$ maps these role definitions to the set of agent names which are the contract parties within the contract document. We identify a set $ContractParties \subseteq AgentNames$ as those agents named within the contract. This means that the following condition must be satisfied:

$\mathbf{\Gamma} \cup CDRoles \cup CDRoleMapping \vdash X$ *where* $X \subseteq ContractParties$.

The qualification on X requires that a contract document only imposes norms on contract parties.

Being an identifiable data item, a contract document, or any element of it, may have additional metadata which may be included in the contract document itself or stored separately. One common piece of metadata associated with most contracts is a contract status. Metadata may be viewed as an additional theory from which the agent can infer information, and is labelled $\Gamma_{metadata}$.

A contract may refer to other contracts, requiring that these additional contracts hold when the referring contract holds. Such additional contracts may also be considered part of the contract's context, and may, among other things, represent societal regulations imposed on the contracting parties.

Contexts (i.e. ontologies, descriptions and regulations imposed on contract parties) may be shared between multiple contracts. Context provides full meaning to the terms, actions and processes described in the contract.

When an agent reasons about a contract, it makes use of the contract's context. Such an agent is said to be operating within the appropriate context. Agents within a common context share a common vocabulary. This implies that each context has to be associated with a domain ontology defining the meaning of the terms used in the interactions. Therefore, the ontology bound to a context must contain at least all the predicates, roles, role hierarchies, actions and processes that are part of its domain.

However, the ontology may not be enough to express the whole context domain semantics. An extra model is needed for dealing with all the aspects of the domain, especially those that are dynamic. We call this the *world model*, which contains sets of conditional rules that use predicates and actions from the ontology. The domain ontology and world model also form part of $\mathbf{\Gamma}$.

Contract Proposal. Until agreed to by the contract parties, a contract document has no normative weight. At this stage, it is referred to as a contract proposal. Once agreed to, the contract becomes binding, and its norms are then considered to come into effect.

4.3 Operational Semantics

So far, we have described the structure of contract documents and norms. Usually, we will be interested not in the semantics of the documents themselves, but how they affect the contract parties, i.e. how, given the evolution of the environment, various norms are instantiated, fulfilled, violated and discharged.

To do this, we now describe the normative environment theory Γ_{NEnv}, which defines predicates that may be used to identify the status of norms as they progress through their lifecycle. A normative environment theory is built around a normative environment,

which is itself a (possibly infinite) sequence of normative states $NS_1, NS_2, \ldots$. Each normative state NS_i in the sequence is defined with respect to the overarching theory $\mathbf{\Gamma}$ (which includes Γ_{NEnv}), and a given set of abstract norms ANS.

Each normative state keeps track of four basic events:

- when an abstract norm is instantiated;
- when an instantiated norm expires;
- when a norm's normative condition holds; and
- when a norm's normative condition does not hold.

In order to formally define a normative state we first define predicates based on an instantiated norm's $NormCondition$ and $ExpirationCondition$ attributes:

Definition 5. ***(The* holds() *Predicate)*** *Let in be an instantiated norm*

$$\langle NormType, NormActivation, NormCondition, \\ NormExpiration, NormTarget\rangle$$

Then, for $N \in \{NormCondition,\ NormExpiration\}$: $holds(in, N)$ evaluates to true if $\mathbf{\Gamma} \vdash N'$, where N' is entailed with all variables in N grounded; otherwise $holds(in, N)$ evaluates to false*.*

Our formal definition of a normative state then identifies those instantiated norms whose normative condition evaluates to true, those whose normative condition evaluates to false, and those whose expiration condition evaluates to true:

Definition 6. ***(Normative State)*** *Let INS be a set of instantiated norms. A normative state NS, defined with respect to a set INS of instantiated norms, and domain environment theory Γ_{Env} and normative environment theory Γ_{NEnv}, is a tuple of the form:*

$$\langle NSTrue, NSFalse, NSExpires\rangle$$

where:

- $NSTrue$ = $\{in \in INS|\ holds(in, NormCondition)$ *is true*$\}$
- $NSFalse$ = $\{in \in INS|\ holds(in, NormCondition)$ *is false*$\}$
- $NSExpires$ = $\{in \in INS|\ holds(in, NormExpiration)$ *is true*$\}$

Since $NSExpires \subseteq NSTrue \cup NSFalse$, it is sufficient to identify the instantiated norms in a normative state, denoted $inst\_norms(NS)$, by the union of those norms whose normative condition evaluates to true, and those whose normative condition evaluates to false. That is to say:

$$inst\_norms(NS) = NSTrue \cup NSFalse$$

Definition 7. ***(Normative Environment)*** *A normative environment NE is a possibly infinite ordered sequence $NS_1, NS_2, \ldots$ where for $i = 1 \ldots$, we say that NS_i is the normative state* prior *to NS_{i+1}.*

Given a normative state, the subsequent normative state is defined by removal of the expired instantiated norms, addition of new instantiated norms, and checking the norm state of all instantiated norms. We therefore define a minimal set of conditions that a normative environment should satisfy:

Definition 8. ***(Normative State Semantics)*** *Let $\mathcal{ANS}$ be an abstract norm store, NE the normative environment $NS_1, NS_2, \ldots$, and for $i = 1 \ldots$, $\mathbf{\Gamma_i}$ a set of* wffs *denoting the domain environment associated with NS_i. For $i = 1 \ldots$, we can define the set of potential norms for NS_i as*

1. *those that are instantiated in the previous state NS_{i-1} ($inst\_norms(NS_{i-1})$)*
2. *those in the abstract norm store that are instantiated w.r.t. $\mathbf{\Gamma_i}$ (i.e., $inst(\mathcal{ANS})$ as defined in Definition 3).*
3. *And not those that have expired in the previous state, i.e., $NSExpires_{i-1}$.*

That is to say, the set of potential norms $PNorms_i$ is defined as follows:

$$PNorms_i = inst\_norms(NS_{i-1}) \cup inst(\mathcal{ANS}) \setminus NSExpires_{i-1}$$

Then $NS_i = \langle NSTrue_i, NSFalse_i, NSExpires_i \rangle$ is defined (as in Definition 6) with respect to the set $PNorms_i$, and theory $\mathbf{\Gamma_i}$.

Clearly, some initial normative state is required, and we define it as

$$NS_0 = \langle NSTrue_0, NSFalse_0, NSExpires_0 \rangle$$

where $NSTrue_0 = \{\}$, $NSFalse_0 = \{\}$ and $NSExpires_0 = \{\}$.

We suggest the following basic set of predicates entailed by Γ_{NEnv}, and in this way characterise how Γ_{NEnv} may be partially specified by the normative environment.[4] In the following definitions we assume a normative environment $\{NS_1, NS_2, \ldots\}$ where $NS_i = \langle NSTrue_i, NSFalse_i, NSExpires_i \rangle$, and $i > 0$. We use the *Gödelisation* operator $\lceil . \rceil$ for naming normative states in the object level language. That is, $\lceil NS_i \rceil$ names normative state NS_i and allows us to use it within *wffs*.

Definition 9. ***(The* instantiated() *predicate)*** *$\Gamma_{NEnv} \vdash instantiated(\lceil NS_i \rceil, in)$ iff $in \in inst\_norms(NS_i)$ and $(in \notin inst\_norms(NS_{i-1}) \vee in \notin NSExpires_{i-1})$. We define by default $\Gamma_{NEnv} \nvdash instantiated(\lceil NS_0 \rceil, in)$.*

Intuitively, $instantiated(NS_i, in)$ holds if the norm in becomes instantiated in NS_i. That is, $instantiated(\lceil NS_i \rceil, in)$ evaluates to true if norm in was instantiated in NS_i, and either was not instantiated in NS_{i-1} or expired in NS_{i-1} (and thus becomes instantiated again in NS_i).

Definition 10. ***(The* expires() *predicate)*** *$\Gamma_{NEnv} \vdash expires(\lceil NS_i \rceil, in)$ if and only if $in \in NSExpires_i$. We also define $\Gamma_{NEnv} \nvdash expires(\lceil NS_0 \rceil, in)$.*

[4] In general, by stating the requirement that some first order theory Γ entails $\phi_1, \ldots, \phi_n$, we are effectively providing a partial specification of Γ. In semantic terms, any model for Γ is also model for $\phi_1, \ldots, \phi_n$.

The $expires()$ predicate holds if an instantiated norm *in* expired within a specific normative state.

Definition 11. ***(The* active() *predicate)*** *$\Gamma_{NEnv} \vdash active(\lceil NS_i \rceil, in)$ if and only if $instantiated(\lceil NS_i \rceil, in)$, or else $(in \in inst\_norms(NS_{i-1}) \wedge in \notin NSExpires_{i-1})$. We also define $\Gamma_{NEnv} \nvdash active(\lceil NS_0 \rceil, in)$.*

$active(\lceil NS_i \rceil, in)$ holds if a norm *in* is instantiated within normative state NS_i. This could be because it was instantiated within that state, or because it was instantiated earlier and has not yet expired.

Definition 12. ***(The* becomesTrue() *predicate)*** *$\Gamma_{NEnv} \vdash becomesTrue(\lceil NS_i \rceil, in)$ iff $in \in NSTrue_i$ and, either $in \in NSFalse_{i-1}$, or $instantiated(\lceil NS_i \rceil, in)$.*

Intuitively, a norm in becomes true in NS_i if its normative condition evaluates to true, and either it was false in state NS_{i-1}, or if not, then in is instantiated in NS_i.

Definition 13. ***(The* becomesFalse() *predicate)*** *$\Gamma_{NEnv} \vdash becomesFalse(\lceil NS_i \rceil, in)$ iff $in \in NSFalse_i$ and, either $in \in NSTrue_{i-1}$ or $instantiated(\lceil NS_i \rceil, in)$.*

Here, $becomesFalse(\ldots)$ is similar to $becomesTrue(\ldots)$, dealing with falsehood rather than truth. The next two predicates check whether a norm is active and true, respectively false, in some normative state.

Definition 14. ***(The* isTrue() *predicate)*** *$\Gamma_{NEnv} \vdash isTrue(\lceil NS_i \rceil, in)$ if and only if $becomesTrue(\lceil NS_i \rceil, in)$, or else, $active(\lceil NS_i \rceil, in)$ and $in \in NSTrue_{i-1}$.*

Definition 15. ***(The* isFalse() *predicate)*** *$\Gamma_{NEnv} \vdash isFalse(\lceil NS_i \rceil, in)$ if and only if $becomesFalse(\lceil NS_i \rceil, in)$, or else, $active(\lceil NS_i \rceil, in)$ and $in \in NSFalse_{i-1}$*

Definition 16. ***(Properties of Γ_{NEnv})*** *$\Gamma_{NEnv} \vdash \neg x$ iff $\Gamma_{NEnv} \nvdash x$. This implies that:*

- *$\Gamma_{NEnv} \nvdash \bot$.*
- *$\neg$ is given a negation as failure semantics.*

Apart from these low level predicates, we may define additional useful predicates. Some of these determine the status of a norm, while others allow access to its operation.

Definition 17. ***(Norm access predicates)*** *Given a norm N with norm type $Type$, activation condition $NormActivation$, expiration condition $NormExpiration$, a norm target set $NormTarget$, and a normative condition made up of a state component $SMaintenanceCondition$ and an action component $AMaintenanceCondition$, the following predicates (which may operate on both abstract and instantiated norms) may be defined:*

$type(N, X) = true$ iff $X = Type$, and false otherwise.
$normActivation(N, X) = true$ iff $NormActivation$ unifies to X, and false otherwise.
$normSCondition(N, X) = true$ iff $SMaintenanceCondition$ unifies to X, and false otherwise.

$normACondition(N, X) = true$ *iff* $AMaintenanceCondition$ *unifies to* X*, and false otherwise.*
$normExpiration(N, X) = true$ *iff* $NormExpiration$ *unifies to* X*, and false otherwise.*
$normTarget(N, A) = true$ *iff there is a unification between some element of* A *and* $NormTarget$*.*

We may define the following predicates based on the normative environment. These predicates form a basis for our normative environment theory Γ_{NEnv}:

Definition 18. ***(the violated() predicate)***
$violated(\lceil NS_i \rceil, in) = normType(in, obligation) \wedge isFalse(\lceil NS_i \rceil, in)$

The fulfilled predicate checks whether a norm has been fulfilled at a specific point in time.

Definition 19. ***(the fulfilled() predicate)***
$fulfilled((\lceil NS_i \rceil, in) = expires(\lceil NS_i \rceil, in) \wedge \neg violated(\lceil NS_i \rceil, in)$
$unfulfilled((\lceil NS_i \rceil, in) = expires(\lceil NS_i \rceil, in) \wedge violated(\lceil NS_i \rceil, in)$

We may also be interested in determining whether a norm is a *violation handler*; that is, if it detects and handles the violation of some other clause. We make the simplifying assumption that a violation handler contains only the $violated()$ predicate in its activating condition.

Definition 20. ***(the violationHandler() predicate)***
$violationHandler(N) = normActivation(N, \lceil violated(X, Y) \rceil)$ *for any* X, Y*.*

Finally, we may want to determine which norm (N_1) is the violation handler for another norm (N_2):

Definition 21. ***(the handlesViolation() predicate)***
$handlesViolation(N_1, N_2) = normActivation(N_1, \lceil violated(X, N_2) \rceil)$ *for any* X

As we will discuss later, we intend to make use of these semantics when evaluating the status of contracts. Before doing so, we illustrate the use of our framework within an example.

5 Example

Building on the Car Insurance Brokerage example presented above, we now show how the normative environment theory evolves, together with its attendant normative states. We assume a contract over the repair of a car has been signed between a repair company ("Bob Repairs") and the broker (Damage Secure). For simplicity, we have directly placed the agent's names within the norm targets, rather than using roles. This contract contains the two clauses[5]:

[5] Due to space constraints, we do not describe the temporal predicates used in the clauses, instead assuming that they are derived from some standard temporal logic.

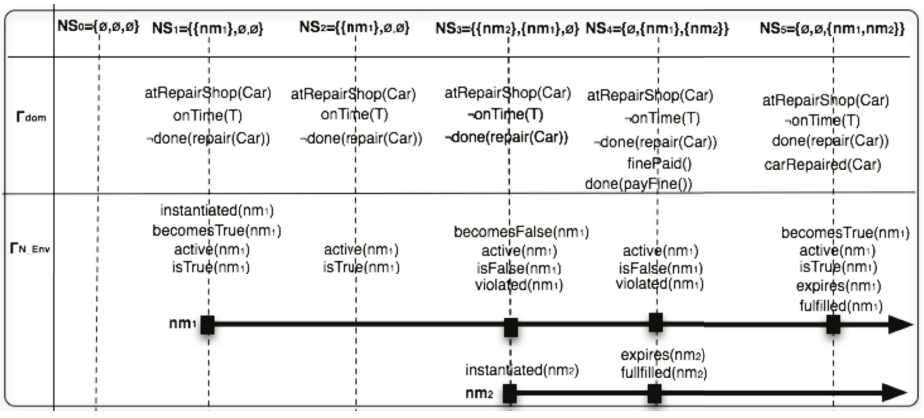


Fig. 1. Domain Environment and Normative Environment lifecycle

The first clause consists of norm nm_1 =

$$\langle obligation, atRepairShop(Car), onTime(T) \wedge atRepairShop(Car), done(repairCar(Car)), BobRepairs \rangle$$

Where $onTime(T) = before(T, contractStartTime + 7days)$. Here, T represents the current time.

The second clause contains norm nm_2 =

$$\langle obligation, violated(nm_1), done(payFine()), finePaid(), BobRepairs \rangle$$

The first clause expresses the obligation imposed on Bob Repairs to repair a car within the first 7 days of the contract start date, provided the car is at the repair shop. The second expresses the obligation on the Repair Company to pay a fine in case it violates the first obligation. At this stage, it must be noted that the obligation on the Repair Company to repair the car holds even if the seven day deadline has passed. Alternatively, if we wanted to model the obligation in such a way that, if after the seven days pass and the car is still not repaired, a fine should be paid and no repair must be made, then we would have to modify the first $NormExpiration$ attribute to read $carRepaired(Car) \vee after(T, contractStartTime + 7days)$.

Figure 1 shows the values of several of predicates belonging to a domain theory Γ_{Dom} ($atRepairShop, onTime, done, finePaid$) and Γ_{NEnv} ($instantiated()$, $violated()$, $expires()$, ...). Some predicates for nm_2, the value of which can easily be inferred, are omitted from the figure, due to limited space.

By default, NS_0 contains empty sets for all its elements. In the next normative state, NS_1, nm_1's $NormCondition$ becomes true, as the car is at the shop. This event causes nm_1 to become instantiated. The norm's $NormCondition$ element evaluates to true (and thus $nm_1 \in NSTrue_1$) until 7 days from the contract start date have passed. At that point (NS_3), the predicate $before(T, contractStartTime + 7days)$

no longer holds, and $NormCondition$ becomes false (and thus $nm_1 \in NSFalse_1$). By definition, this means that the violated() predicate for nm_1 evaluates to true. This causes the instantiation of the second norm, as seen by its $NormActivation$ parameter. By paying the fine at the next normative state NS_4, the repair company fulfils nm_2's $NormExpiration$ condition, and this brings nm_2 to a $fulfilled()$ state. Finally, we assume that the car is repaired at NS_5, meaning that $violated()$ evaluates to false for nm_1. This means that nm_1 is also fulfilled, as its $NormExpiration$ now evaluates to true.

6 Discussion and Conclusions

The normative framework we have described fulfils all of the requirements described earlier. Not only are we able to determine whether a violation took place (via the $violation(\ldots)$ predicate), but we may also detect the occurrence of additional *critical states* at which some normative event related state change took place. These critical states correspond to the various predicates described above. Additional, domain dependent critical states may be defined using the information found within the normative environment. Verification of a normative system may be performed by forward simulation over the domain and normative environments.

We assume that any norm aware agent capable of being affected by norms, is associated with its own normative environment (and resulting normative environment theory). In fully observable environments, each agent's theory would be identical, but in other domains, these theories may diverge. In the context of contracting, we assume that the contract may state which agent's theories should be used when determining whether penalties (or rewards) should be imposed.

Our model does not describe what should occur if an obligation is violated. Instead, we assume that agents make use of the normative model to undertake their own practical reasoning. An agent may determine which norms affect it at any stage, and base its decisions on these.

One interesting aspect of our model (as illustrated by norm nm_1 in the example) is that norms may be violated for a certain period of time, after which they may return to an *unviolated* state. This is particularly useful when dealing with contracts, as penalties may be assessed over the duration of a violation, with the norm still having normative force over an agent. This differs from the way most deontic theories deal with norms (for example [7]).

While a large variety of electronic contracting languages exist, many only specify an informal [8], or programming language based [3] semantics. Conversely, formal languages, such as LCR [4] have limited expressibility. In contrast, our approach of defining a rich contracting language, and then constructing its semantics, is intended to overcome these weaknesses.

The work presented here has been inspired by a number of other researchers. For example, [9] described the use of *landmarks* as abstract states which "are defined as a set of propositions that are true in a state represented by the landmark". These landmarks may thus be seen as similar to critical states. The framework described by [10] shares some similarities with our approach. Their focus on sanctions (which, in our model,

are implemented via additional norms) means that they only allow for very specific, predefined normative states, and that violations in their framework may only occur once.

[11] described a predicate based event calculus approach to keeping track of normative state in contracts. However, their work focused on specifying an XML based representation of event calculus, and made use of event calculus primitives to specify their contracts, resulting in a very unwieldy and unrealistic contract representation, and very few norm related predicates. Finally, [12] showed how petri-nets could be used to perform contract monitoring, but her representation is best suited for those contracts which can be expressed as workflows.

In this paper, we have presented the normative underpinnings of our contract model, showing how norms are represented, and how we can determine their state as the environment changes. We have successfully migrated this approach to the contract level, allowing us to identify the state of a contract (e.g. drafted, active, etc.) at any point in time. IST-CONTRACT intends to create an entire contracting ecosystem, and we are currently using the normative model to define the behaviour of various contract-supporting components of the system such as the contract store and contract managers. In the near-term, we intend to see whether we can migrate the semantics described here to additional levels of the framework, for example to specify the form of inter-agent protocols. In the long-term, we are interested in examining how norm-conflict mechanisms may best make use of the framework, and are also looking at the effects of partial, and conflicting information on the semantics.

Acknowledgements

This work has been funded mainly by the FP6-034418 IST-CONTRACT project. Javier Vázquez-Salceda's work has been also partially funded by the Ramón y Cajal program of the Spanish Ministry of Education and Science. All the authors would like to thank the CONTRACT project partners for their inputs to this work.

References

1. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Wooldridge, M., Sierra, C.: Automated negotiation: Prospects methods and challenges. Group Decision and Negotiation 10(2), 199–215 (2001)
2. Grosof, B., Poon, T.C.: SweetDeal: Representing agent contracts with exceptions using semantic web rules, ontologies, and proceedings descriptions. International Journal of Electronic Commerce 8(4), 61–98 (2004)
3. Kollingbaum, M.: Norm-governed Practical Reasoning Agents. PhD thesis, University of Aberdeen (2005)
4. Dignum, V., Meyer, J.J., Dignum, F., Weigand, H.: Formal specification of interaction in agent societies. In: Proceedings of the Second Goddard Workshop on Formal Approaches to Agent Based Systems, pp. 37–52 (2002)
5. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Resolving conflict and inconsistency in norm-regulated virtual organizations. In: Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems, Honolulu, Hawaii, USA, pp. 632–639 (2007)

6. Oren, N., Luck, M., Miles, S., Norman, T.J.: Argumentation inspired heuristics for resolving normative conlict. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428. Springer, Heidelberg (2008)
7. van der Torre, L.: Contextual deontic logic: Normative agents, violations and independence. Annals of Mathematics and Artificial Intelligence 37(1-2), 33–63 (2003)
8. Milosevic, Z., Dromey, R.G.: On expressing and monitoring behaviour in contracts. In: Proceedings of the Sixth International Enterprise Distributed Object Computing Conference, pp. 3–14 (2002)
9. Dignum, V.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD thesis, Universiteit Utrecht (2004)
10. Fornara, N., Colombetti, M.: Specifying and enforcing norms in artificial institutions. In: Normative Multi-agent Systems. Number 07122 in Dagstuhl Seminar Proceedings (2007)
11. Farrell, A.D.H., Sergot, M., Salle, M., Bartolini, C.: Using the event calculus for tracking the normative state of contracts. International Journal of Cooperative Information Systems 4(2-3), 99–129 (2005)
12. Daskalopulu, A.: Modelling Legal Contracts as Processes. In: Eleventh International Conference and Workshop on Database and Expert Systems Applications, pp. 1074–1079 (2000)

An Automata-Based Monitoring Technique for Commitment-Based Multi-Agent Systems

Paola Spoletini[1] and Mario Verdicchio[2]

[1] Università dell'Insubria, Como, Italy
paola.spoletini@uninsubria.it
[2] Università degli studi di Bergamo, Italy
mario.verdicchio@unibg.it

Abstract. In open multi-agent systems (MASs) we cannot assume agents to be developed in a centralized fashion. Recent proposals of commitment-based communication frameworks aim at increasing such openness. Interaction with agents whose behavior does not follow a universal standard raises the need for some means of protection for each agent. In this work we propose an automata-based monitoring module that continuously supports an agent during its life in a MAS. Such module includes a *Word Composer* that observes exchanged messages and keeps track of significant past interactions to express an agent's input in the form of time-stamped words, and a *Word Analyzer* that processes such words and matches them against some properties expressed in linear temporal logic which are supposed to hold throughout the interactions.

1 Introduction

Communication may be considered as playing a fundamental role in increasing the openness of multi-agent systems (MASs), as interaction standards that need not take agents' internal architecture into account allow for systems populated by heterogeneous, independently developed entities.

The most significant agent communication language (ACL) standard proposed so far, FIPA ACL, despite some minor changes throughout the years, has always been providing mental-state-based specifications. Among the issues that rise from such an approach, the most compelling is probably the fact that programmers are supposed to create software with a specific architecture implementing such prescribed mental states.

To counter this limitation to MASs' openness, some researchers have proposed ACL standards with a commitment-based semantics [4, 14], according to which every communicative act is seen as the creation or the modification of a commitment binding the agent to the others. While mental states are subjective and private, commitments are objective and public, and can be stored in public records for further reference.

These advantages come with a cost, dealing with different aspects of a commitment based open MAS. Firstly, if agent interaction is expressed in terms of public commitments, every agent needs to check whether such commitments are

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 172–187, 2009.

fulfilled or not on the basis of the events that have occurred in the system. We might assume such task to be performed by a centralized service to which all agents subscribe, but as one of the commitment proposal's main aims is to increase openness in MASs, we might as well prescribe that each agent be provided with a monitoring module. Moreover, openness means also that an agent cannot rely on any assumption about the agents it interacts with other than a common communication framework.

This raises the need for a way to protect the agent from potentially harmful interactions which might end up with contradictory commitments (which would inevitably lead to a violation of one of them) or with commitments to actions that are not compatible with the agent's characteristics or resources. The agent's monitoring module, thus, would also help keep its interactions in an open system safe, by checking whether the events are compatible with the properties that are part of the agent's specifications.

We propose that such a module be implemented as a component comprised of two submodules: a *Word Composer* elaborates exchanged messages in the form of timestamped words, which are in turn processed by a *Word Analyzer* that, exploiting finite state automata on infinite words, analyzes the state of the MAS and checks whether some properties, expressed in linear temporal logic, hold.

We present our proposal in the remainder of this work, which is organized as follows. Section 2 provides the theoretical background about the concepts supporting commitment-based agent interactions and the monitoring module, which is illustrated in detail in Section 3; in Section 4 some of the most significant related work is referred to; finally, Section 5 concludes.

2 Background

Let us start with some linear temporal operators, their definitions, and some abbreviations, followed by a suitable content language to represent the commitment-based domain the agents are working in. Then, we illustrate the theoretical background on Büchi and alternating automata, which our monitoring module is based on.

2.1 LTL$^{\pm}$

The monitored properties are expressed using the notion of time as introduced by Linear Temporal Logic with both past and future modalities (LTL$^{\pm}$) [12]. In the following we describe a propositional version of LTL, supposing that the atomic propositions belong to a specific Content Language (CL). LTL$^{\pm}$ is a modal logic in which modalities refer to time and, considered a set of atomic propositions CL, its syntax is given over it in BNF as follows:

$$\varphi ::= true|p|\neg\varphi|\varphi \vee \varphi| \mathsf{X}\varphi \,|\mathsf{G}^{+}\varphi \,|\mathsf{F}^{+}\varphi \,|\mathsf{Until}(\varphi,\varphi)$$
$$\mathsf{P}\varphi \,|\mathsf{G}^{-}\varphi \,|\mathsf{F}^{-}\varphi \,|\mathsf{Since}(\varphi,\varphi) \mid \mathsf{WUntil}(\varphi,\varphi) \mid \mathsf{Z}^{+}(\varphi,\varphi)$$

where the modal operators X, P, F^{+}, F^{-}, G^{+}, G^{-}, Until, Since, WUntil, Z^{+} are called *next*(time), *previous*(time), *eventually in the future*, *eventually in the past*,

always in the future, *always in the past*, *until*, *since*, *weak until* and *until and no longer*, respectively. The boolean operators $\wedge$ and $\Rightarrow$ are obtained, as usual, by composing $\vee$ and $\neg$.

The semantics of LTL$^{\pm}$ is given on a Kripke structure M, that consists of a tuple $\langle S,\ R,\ L\rangle$, where S is a finte set of states, $R \subseteq S \times S$ is the transition relation, and $L : S \rightarrow 2^{CL}$ is a labeling function that labels each state with the propositions in CL that are true in that state. A path $\pi = s_0,\ s_1,\ \ldots$ is an infinite sequence of states in S such that, $\forall i \geq 0$, $(s_i, s_{i+1}) \in R$.

We give the semantics of LTL$^{\pm}$ formula on a Kripke structure M using the following notation. Let π^i in a path $\pi = s_0\ s_1, \ldots$ be the suffix of π that starts from s_i. If φ is a formula, $M,\ \pi \models \varphi$ means that φ holds in the state initial state s_0 of the path π in the Kripke structure M. The relation $\models$ is then defined inductively as follows:

- $M,\ \pi \models\ p$ iff $p \in L(s_0)$;
- $M,\ \pi \models\ \neg\varphi$ iff $M,\ \pi \not\models \varphi$;
- $M,\ \pi \models\ \varphi_1 \vee \varphi_2$ iff $M,\ \pi \models \varphi_1$ or $M,\ \pi \models \varphi_2$;
- $M,\ \pi \models \mathsf{X}\varphi$ iff $M,\ \pi^1 \models \varphi$;
- $M,\ \pi \models \mathsf{Until}(\varphi_1, \varphi_2)$ iff there exists $k > 0$ such that $M,\ \pi^k \models \varphi_2$ and, for all $0 \leq j < k$, $M,\ \pi^j \models \varphi_1$;
- $M,\ \pi \models \mathsf{P}\varphi$ iff there is a path π_* s. t. $\pi_*^1 = \pi$ and $M,\ \pi_* \models \varphi$;
- $M,\ \pi \models \mathsf{Since}(\varphi_1, \varphi_2)$ iff there is a path π_* s. t. $\pi_*^n = \pi$ and there is a $0 \leq k < n$ such that $M,\ \pi_*^k \models \varphi_2$ and, for all $0 \leq j < k$, $M,\ \pi_*^j \models \varphi_1$;

The rest of the temporal modalities can be expressed using the ones above as follows: $\mathsf{F}^+\varphi{=}\mathsf{Until}(\mathit{true}, \varphi)$, $\mathsf{G}^+\varphi{=}\ \neg\mathsf{F}^+\neg\varphi$, $\mathsf{F}^-\varphi{=}\mathsf{Since}(\mathit{true}, \varphi)$, $\mathsf{G}^-\varphi{=}\ \neg\mathsf{F}^-\neg\varphi$, $\mathsf{WUntil}(\varphi_1, \varphi_2){=}\mathsf{G}^+\varphi_1 \vee \mathsf{Until}(\varphi_1, \varphi_2)$, (without a mandatory occurrence of φ_2) $\mathsf{Z}^+(\varphi_1, \varphi_2){=}\mathsf{WUntil}(\varphi_1, \varphi_2) \wedge \mathsf{G}^+(\varphi_2 \Rightarrow \mathsf{G}^+\neg\varphi_1)$.

Notice that the operators X and P give also a quantitative notion of time, since they can identify temporal instants in the domain of natural numbers.

Given an integer $K > 1$, we allow also a more concise form to express the boolean combination of nested X (and P, respectively), in the following way:

- $\mathsf{X}^K\varphi$ ($\mathsf{P}^K\varphi$) stands for K nested $\mathsf{X}\varphi$ ($\mathsf{P}\varphi$) operators.
- $\mathsf{F}^+_{\bullet K}\varphi$ ($\mathsf{F}^-_{\bullet K}\varphi$) with $\bullet \in \{<, \leq\}$ stands for
  $\varphi \vee \mathsf{X}\varphi \vee \ldots \vee \mathsf{X}^{K-1}\varphi$ ($\varphi \vee \mathsf{P}\varphi \vee \ldots \vee \mathsf{P}^{K-1}\varphi$) or
  $\varphi \vee \mathsf{X}\varphi \vee \ldots \vee \mathsf{X}^{K}\varphi$ ($\varphi \vee \mathsf{P}\varphi \vee \ldots \vee \mathsf{P}^{K}\varphi$)
  for $\bullet = <$ and $\bullet = \leq$ respectively.
- $\mathsf{G}^+_{\bullet K}\varphi$ ($\mathsf{G}^-_{\bullet K}\varphi$) with $\bullet \in \{<, \leq\}$ stands for
  $\varphi \wedge \mathsf{X}\varphi \wedge \ldots \wedge \mathsf{X}^{K-1}\varphi$ ($\varphi \wedge \mathsf{P}\varphi \wedge \ldots \wedge \mathsf{P}^{K-1}\varphi$) or
  $\varphi \wedge \mathsf{X}\varphi \wedge \ldots \wedge \mathsf{X}^{K}\varphi$ ($\varphi \wedge \mathsf{P}\varphi \wedge \ldots \wedge \mathsf{P}^{K}\varphi$)
  for $\bullet = <$ and $\bullet = \leq$ respectively .

Notice that we are working using natural numbers as domain and, in this scenario, past modalities do not add any expressive power with respect to classical LTL [11], but allow us to represent the required properties in a shorter and more elegant fashion.

The LTL$^{\pm}$ temporal operators may be considered as the domain-independent part of the language that allows for the description of the properties an agent needs to monitor, while the context of the MAS where the agent is running determines the domain the propositional atoms refer to. As our proposal deals with MASs with a commitment-based communication framework, let us briefly provide a language which is expressive enough to describe such type of interaction. In this perspective, a message exchange is viewed as an action performed by an agent to create or modify the commitments that bind it to other agents.

Events are reified, and each event token belongs to at least an event type. An event brought about by an agent is called an action, and we write $Done(e, x, \tau)$ to mean that event e of type τ is brought about by agent x. We use the "m-dash" character as a shorthand for existential quantification. For instance, $Done(e, -, \tau)$ is defined as $\exists x Done(e, x, \tau)$.

A commitment is a social state between agents comprised of four components: an *event* e that has created the commitment, a *debtor* x which is the agent who is committed, a *creditor* y which is the agent the debtor is committed to, and a *content* u which represents the state of affairs the debtor is committed to bring about, and the relevant predicate is $Comm(e, x, y, u)$. By including event e in the parameters of a commitment, we make it linkable to the event that generated it for further reference. As we do not intend to depart from classical first-order logic, we let the content of a commitment be represented by a term u, and we write $\lfloor u \rfloor$ to refer to the relevant LTL$^{\pm}$ formula.

Intuitively, a commitment is fulfilled when its content, or, more precisely, the LTL$^{\pm}$ formula corresponding to its content is true, and is violated when its content is false. Allowing for more expressive contents including commitments themselves would easily lead to situations in the likes of the "liar paradox", which are far from automatically manageable. Investigating the allowable extent of content language expressiveness lies beyond the scope of this work. In our view, an agent's monitoring module gathers the truth values of the propositional atoms included in a commitment's content $\lfloor u \rfloor$ at all the needed states, as prescribed by the temporal operators in $\lfloor u \rfloor$. As soon as the module is able to calculate a truth value for the whole formula, the agent knows whether the relevant commitment has been fulfilled or violated. This process is described in detail in the next section.

Agents create commitments by performing suitable tokens of *commitment manipulation* action types, like make commitment (mc). The reader may refer to [18] for further details about commitment manipulation. The effects of the performance of a commitment manipulation action are illustrated in the form of axioms. The scope of the validity of formulae is limited to the class of LTL$^{\pm}$ models that fulfill the constraints imposed by such axioms. For instance, if an agent (not necessarily x or y) performs an action of making a commitment with x as debtor, y as creditor, and u as content, then the relevant commitment holds until it is either fulfilled, or violated, after which it no longer exists:

$$Done(e, -, mc(x, y, u)) \Rightarrow Comm(e, x, y, u) \mathsf{Z}^{+} Fulf(e, x, y, u) \vee Viol(e, x, y, u).$$

In many cases, a possible content language CL that allows for a significant description of a domain must exploit first-order logic's expressiveness. However, to

be able to write the properties to be monitored in the form of LTL$^{\pm}$ formulae, we need to translate the first-order logic sentences into propositions. Current propositional encodings (naive propositionalizations) result in extremely large propositional encodings even for moderate applications. No more efficient solution has been found yet, even though some promising proposals can be found in the literature [13]. For our purposes in this work, we assume that the monitoring agent lives in a system where at each interaction a finite domain is in place, and all the relevant identifiers are agreed upon by the participating agents. This allows for the above-mentioned propositional encodings.

2.2 Finite Automata on Infinite Words

Let us remind here the definitions of classical and alternating Büchi automata (BAs [16] and AAs [3], respectively), which consitute the basis for our monitoring module. Formally, a BA A is a tuple $\langle \Sigma, S, s_0, \delta, F \rangle$, where: Σ is the finite set of the input symbols, S is a finite set of states, $s_0 \in S$ is the initial state, $\delta : S \times \Sigma \to 2^S$ is the transition relation, and $F \subseteq S$ is the set of accepting states. Differently from classical finite state automata on finite words, these automata will accept infinite words, using the Büchi condition, i.e., a word $w = a_0, a_1, \ldots$ (for each $i \geq 0$, $a_i \in \Sigma$) is accepted by a BA A if exists an infinite sequence $s = s_0 s_1 s_2 \ldots$ where, s_0 is the initial state, for each $i \geq 0$, $\delta(s_i, a_i) = s_{i+1}$ and at least one state $s \in F$ appears on π infinitely many times.

AAs can be defined as BAs with the only difference in the transition function that becomes $\delta : S \times \Sigma \to B^+(S)$, where $B^+(S)$ is the positive boolean combination of the elements in S, i.e., a boolean combination using $\wedge$ and $\vee$ but not $\neg$. These automata allow two modalities: nondeterminism, also called existential modality, and parallelism, also called universal modality. As seen in the definition above, the former, given an input letter, allows the automaton to fire the transition choosing where to move among different possible targets. A word is accepted if at least one of these alternatives generates an acceptance run. Symmetrically, the latter makes the automaton move with just one transition in more than one state. This can be seen as the creation of as many copies of the automaton as the states to reach with a universal branch. In this case a word is accepted if it is accepted by all the generated copies.

LTL only with future modality is strongly correlated to BA and AA [3]. In the following we will show how we exploit this correlation.

3 The Monitoring Module

Figure 1 provides an overview of the monitoring module we are proposing. It is comprised of two submodules: the Word Composer (WCS) and the Word Analyzer (WAS), which support an agent during its interactions in the MAS. The WCS includes a sniffing functionality to get a copy of every message that the agent exchanges. Among the sniffed messages, only those dealing with the atomic propositions that appear in the monitored property are selected. Not only the

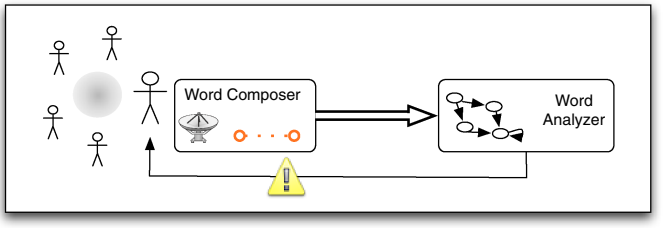


Fig. 1. The monitoring module

WCS processes the received data to prepare the input for the next component, but, should the monitored property contain past tense operators, the WCS is also in charge of keeping track of the relevant information for future evaluation of the truth value of formulae of this kind. Subsection 3.2 provides more details about the construction of the input in the form of a time-stamped word and the processing of past-directed temporal operators. Once the input is ready, it is sent to the WAS, which consists of an alternating automaton functioning as a language acceptor. An unaccepted WCS word means that the state of the MAS does not satisfy the property expressed by the monitored formula, and such violation is notified to the agent. In accordance with the criticality level of the task the agent is supposed to carry out, it will consider the notification from the WAS as a warning or it will abandon the MAS.

3.1 Managing Temporal Aspects

$\text{LTL}^{\pm}$ allows for the creation of formulae with an arbitrary nesting of past and future tense operators. Nevertheless, the two temporal modalities can be processed separately. Gabbay [8] shows that a formula with nested operators can always be algorithmically broken down into subformulae only with future- and past-directed operators. This procedure is performed at a cost of a non-elementary blow up in the number of nested alternated modalities, and this complexity may result in a significant impact on the dimensions of the automaton needed to monitor the formula in worst case scenarios.

However, our approach mainly addresses formulae which are already separated or have a rather small number of nestings, as these are easier to relate with. Gabbay illustrates this technique by providing eight fundamental rules that are to deal with all the possible nesting combinations of the Since and Until operators, which are taken by the author as primitive. By exploiting the relations between these two operators and all the others, we can elaborate similar rules for our temporal language, which may be implemented as a formula preprocessor.

Moreover, as illustrated in more detail in Section 3.2, past operators that do not include any future operator can be processed into simple propositional atoms. In fact, with the evolving of time and the changing of data, the past can be iteratively evaluated on the basis of the previous input, hence its truth value can always be available at the present time as the value of propositional letters.

This means that we do not need to separate formulae in which past operators are embedded in future operators, but only need to take care of the opposite nesting (future operators nested in past operators).

Let us introduce an example of a simple formula to illustrate this approach: $\mathsf{G}^+(\mathsf{F}^-\mathsf{F}^+a \Rightarrow \mathsf{F}^+_{<10}b)$. In this formula the future tense operator F^+ is placed within the scope of the past tense operator F^-. By exploiting the well known equivalences $\mathsf{F}^+a \Leftrightarrow \mathsf{Until}(true,a)$ and $\mathsf{F}^- a \Leftrightarrow \mathsf{Since}(true,a)$ and Gabbay's rule

$$\begin{aligned}\mathsf{Since}(q,p\wedge \mathsf{Until}(B,A)) \Leftrightarrow\ & (\mathsf{Since}(q,p) \wedge \mathsf{Since}(B,p) \wedge B \wedge \mathsf{Until}(B,A)) \vee \\ & (A \wedge \mathsf{Since}(B \wedge q,p)) \vee \\ & (\mathsf{Since}(q,A \wedge q\wedge \mathsf{Since}(B,p) \wedge \mathsf{Since}(q,p))),\end{aligned}$$

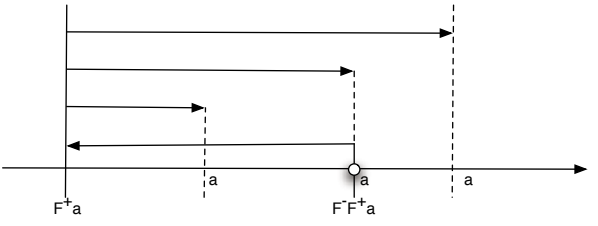


Fig. 2. Possible models for $\mathsf{F}^-\mathsf{F}^+a$

By matching q, p, and B with $true$, and A with a, and by equivalences $\mathsf{Since}(true,true) \Leftrightarrow true$ and $true \wedge p \Leftrightarrow p$, we obtain

$$\mathsf{F}^-\mathsf{F}^+a \Leftrightarrow \mathsf{Until}(true,a) \vee a \vee \mathsf{Since}(true,a) \Leftrightarrow \mathsf{F}^+a \vee a \vee \mathsf{F}^-a.$$

More intuitively, $\mathsf{F}^-\mathsf{F}^+a$ holds at present time t_p if and only if there exists a $t < t_p$ where F^+a holds, which means that there must exist a $t' > t$ where a holds. Nothing is said about t' with respect to t_p, so that we could have $t' > t_p$, $t' = t_p$, or $t' < t_p$, which correspond to the three subformulae in the disjunction we obtained by Gabbay's rule. Some models for $\mathsf{F}^-\mathsf{F}^+a$ are depicted in Figure 2, while a model for the complete example is in Figure 3.

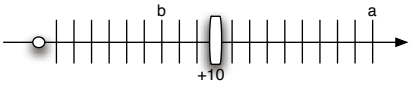


Fig. 3. A sequence of events that satisfies the example

The outmost G^+ operator is not processed as part of the formula like the other operators, but taken as a procedural directive that prescribes that the monitoring task be continually performed.

3.2 The Word Composer

The input alphabet on which our monitoring module works is composed by propositions, which are either sniffed CL formulae translated into propositions or past subformulae evaluated on the basis of previously sniffed data and eventually flagged with a truth value. As our temporal model has a starting point, the input words are infinite on the right (in the future) but finite on the left (in the past), which means that a past subformula always relies on a finite support. These considerations have helped to prove the set of past-directed LTL$^{\pm}$ subformulae to be a language accepted by a deterministic BA[1] [15]. Our monitoring module exploits these results and relies on deterministic BAs to compute the truth value of past subformulae. In the following, we omit to represent these BAs and assume that the results of their computation is kept in a finite memory, which means that past subformulae can be treated as simple propositions.

Let us focus on the unbounded operator $\mathsf{Since}(\varphi, \psi)$, with φ and ψ either present or past formulae (since we are working under the hypothesis of a separated form, we can ignore future-directed examples).The formula is true at a certain instant if in the past ψ was true and since then φ has been true. In order to evaluate this formula at the present time, we would need to keep track of all the previous literals back to the first ψ or, even worse, back to the origin of the system. This problem can be overcome by evaluating Since at all instants, also if its truth value is not immediately required, for future evaluation in combination with other propositions describing the states of the system.

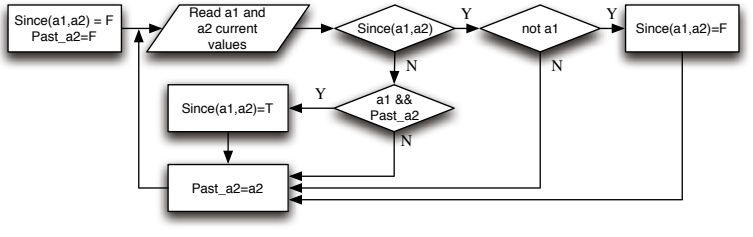


Fig. 4. Flow Diagram of the algorithm to evaluate Since(a1, a2)

More precisely, the truth value of $\mathsf{Since}(\varphi, \psi)$ is evaluated using the algorithm represented by the flow diagram in Figure 4. An analogous function can be directly defined for G^- and F^- or, alternatively, these operators can be expressed in terms of Since.

A difference between the evaluation of formulae with bounded and unbounded operators is that in the former case not only the truth values of the arguments, but also the simple ticking of time influences the final result. To evaluate a bounded operator, in fact, we need a counter that is activated when the argument of the operator becomes true and expires after as many instants as indicated

[1] A deterministic BA is a BA such that the transition function is limited: $\delta : S \times \Sigma \rightarrow S$.

by the numerical constant accompanying the bounded operator. For instance, consider $\mathsf{F}^{-}_{<K}(a)$. The evaluation procedure is activated when a is sniffed and it evaluates $\mathsf{F}^{-}_{<K}(a)$ to true until K time instants have passed without a being detected to hold again.

Taking these considerations into account, the example $\mathsf{G}^{+}((a \wedge \mathsf{F}^{-}c) \rightarrow \mathsf{F}^{+}_{<10}b)$ can be seen as $\mathsf{G}^{+}((a \wedge \texttt{evalF(c)}) \rightarrow \mathsf{F}^{+}_{<10}b)$, with `evalF(c)` computed using the algorithm represented in Figure 5.

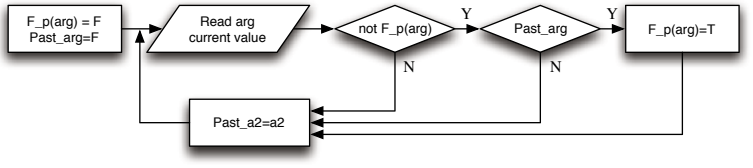


Fig. 5. Flow Diagram of the algorithm to evaluate F^{-}(arg)

3.3 The Word Analyzer

To illustrate how the MWA works, let us first focus on its input. The words composed as explained in Section 3.2 are timestamped, and the temporal distance between two adjacent literals (i.e. the numerical difference between the relevant timestamps) is not constant, but depends on when each data item has been sniffed. Our logic, on the contrary, is based on a model with a uniform distribution of time between successive states. To overcome this lack of temporal uniformity between the events that create the literals in the word and the flow of time in the system, we define an information preserving *filling procedure*. The basic idea is the following: under the hypothesis of choosing a small enough time unit[2], at each instant the atomic propositions are assigned the value they had the last time they were sniffed. Before the first sniffing, the propositions are assigned an initialization value. Notice that past operators are considered as atomic propositions, hence, the changing of their evaluation is equivalent to a new sniffed value.

Figure 6 illustrates the filling procedure. The channel attached to the antenna carries the sniffed values of the propositions, which fill the word (the linear structure below the channel) until the successive sniffing. Under this hypothesis, our input words are coherent with the AA model presented in Section 2.2. As already mentioned, a strong correlation between AA and LTL exists which can be effectively exploited after the past subformulae elimination process (see Section 3.2) turns the monitored LTL$^{\pm}$ formula into an LTL sentence. A methodology to translate an LTL formula into an AA is presented in [17]. It consists in interpreting the subformulae with temporal operators as states, and the formula itself as the initial state, and in defining a particular transition function

[2] The time unit has to be small enough to guarantee no two changes of any atomic proposition values occur in the same time unit.

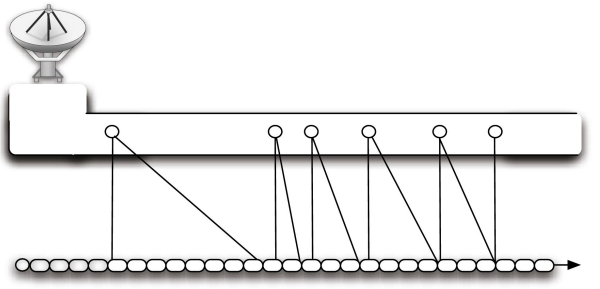


Fig. 6. An example of word filling

that preserves the composition relations between subformulae. We can exploit this approach for our monitoring purposes, but the following drawback must be taken into account. The concise operators built with a boolean combination of nested X operators lead to as many subformulae as the elements in the set we obtain from the transitive closure of the relation of being a subformula of the initial one. This may cause the construction of an AA with a large number of states, negatively impacting on the monitoring module's performance. To overcome this limitation, we follow the approach based on Alternating Modulo Counting Automata (AMCAs), AAs enriched with a finite set of finite counters, as proposed in [15].

An AMCA is a tuple $\langle \Sigma, S, s_0, \mu, \delta, F \rangle$, where: Σ is a finite set of the input symbols, S is a finite set of states, $s_0 \in S$ is the initial state, μ is a positive integer such that $Cnt = [0, \ldots, \mu]$ is a finite set of finite counters, $\delta : S \times \Sigma \times Cnt \rightarrow B^+(S \times Cnt)$ is the transition relation, $F \subseteq S$ is the set of accepting states. These automata do not add any expressive power to AAs, but they allow for a more concise representation of bounded operators. Thus, following the idea illustrated in [17], an LTL formula with metric operators can be translated into an AMCA with an algorithm which is depicted in the form of pseudo-code in Figure 7.

For the sake of simplicity, we suppose that all negations have been pushed to the most inner level of the atomic propositions using well-known transformation procedures. The basic idea is that the algorithm builds as many states of the automaton as the number of temporal subformulae in the formula, possibly including the formula itself, which corresponds to the initial state, and two additional states, *false* and *true*. Function TransitionFrom(s,p,s.Counter) computes the transitions from state s labelled with p and associated with value of the counter at state s. Function Non_det_Branch(⟨s1,assignmentOn (s1.Counter)⟩, ⟨s2,assignmentOn(s2.Counter)⟩) (Universal_Branch(⟨s1,assignmentOn(s1.Counter)⟩, ⟨s2,assignmentOn(s2.Counter)⟩)) creates a nondeterministic (universal) branch between two transitions from the current state s to s1 and s2, respectively. Both transitions are labeled with the relevant counter value assignments.

Let us consider an example with an agent joining a MAS with a service provider. Before the agent's deployment, we can only verify the properties which deal solely with the agent's behavior. Once the agent joins the MAS, as we cannot

```
Automaton(Phi){
   for all temporal subformulae f of Phi
      new_state(f);
   new_state(false);
   new_state(true);
   set_initial_state(Phi);
   for all states s
      if(s has the form Until)
         set_final_state(s);
   Phi.Counter=0;
   put Phi in reachedState;
   for all states s in reachedState
      for all propositions p in Phi
         TransitionFrom(s,p,s.Counter) goes to
         PositiveBooleanCombination((st,assignmentOn(st.Counter)));
         delete s from reachedState;
         put s in processedState;
         for all st in PositiveBooleanCombination((st,assignmentOn(st.Counter)))
            if(st not in processedState && st!=false && st!=true)
               put st in reachedState;
  }

TransitionFrom(Phi,proposition,Phi.Counter){
   if(Phi == s1 OR s2)
       TransitionFrom(Phi,proposition,Phi.Counter)=
       Non_det_Branch(TransitionFrom(s1,proposition,Phi.Counter),
                      TransitionFrom(s2,proposition,Phi.Counter));
   if(Phi == s1 AND s2)
      TransitionFrom(Phi,proposition,Phi.Counter)=
      Universal_Branch(TransitionFrom(s1,proposition,Phi.Counter),
                       TransitionFrom(s2,proposition,Phi.Counter));
   if(Phi is an atom)
      if(Phi == proposition)
         TransitionFrom(Phi,proposition,Phi.Counter) goes to (true, Phi.Counter=0);
      else
         TransitionFrom(Phi,proposition,Phi.Counter) goes to (false, Phi.Counter=0);
   if(Ph == Until(s1,s2))
      TransitionFrom(Phi,proposition,Phi.Counter)=
      Non_det_Branch(TransitionFrom(s2,proposition,Phi.Counter),
                     Universal_Branch(TransitionFrom(s1,proposition,Phi.Counter),
                                      (Phi,no_assignment)));
   if(Phi=X(s,K))
      if(Phi.Counter<K && Phi.Counter+inc(t)<K)
         TransitionFrom(Phi,proposition,Phi.Counter) goes to (Phi, Phi.Counter+=inc);
      if(Phi.Counter==K || Phi.Counter+inc(t)>K)
         TransitionFrom(Phi,proposition,Phi.Counter) goes to (s, s.counter=0);
}
```

Fig. 7. Pseudo-code for the translation of a formula ϕ into an AMCA. X(s,K) stands for $\mathsf{X}^K(s)$.

make any assumption on the behavior of the other agents, we need to rely on the monitoring module to ensure that also more general properties involving all the entities in the MAS hold. For instance, we may be interested in the fact that all the commitments made by the service provider *sp* towards our agent x are fulfilled within 10 time units:

$$\mathsf{P}(\neg Comm(-, sp, x, u)) \wedge Comm(e, sp, x, u) \Rightarrow \mathsf{F}^{+}_{<10}(Fulf(e, sp, x, u)),$$

where $\lfloor u \rfloor = \phi$. Taking the definition of fulfillment into account, after the propositionalization of the formulae we obtain:

$$C_u \wedge \mathsf{P}(\neg C_u) \Rightarrow \mathsf{F}^{+}_{<10}(F_u) \text{ and } F_u \Leftrightarrow \phi,$$

that can be substituted by $C_u \wedge \mathsf{P}(\neg C_u) \Rightarrow \mathsf{F}^{+}_{<10}(\phi)$. For the sake of simplicity we assume ϕ to be a propositional letter $b \in$ CL. The task then boils down

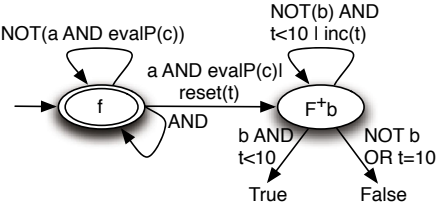


Fig. 8. The AMCA for the formula in the example. AND states for $\wedge$, OR for $\vee$, NOT for $\neg$, f is the formula $\mathrm{G}^{+}((a \wedge \mathrm{evalP(c)}) \rightarrow \mathrm{F}^{+}_{<10}b)$, and t is the counter associated with $\mathrm{F}^{+}_{<10}b$.

to monitor $(\mathsf{G}^{+}((a \wedge \texttt{evalP(c)}) \rightarrow \mathsf{F}^{+}_{<10}b))$, by means of the relevant AMCA, depicted in Figure 8.

The automaton has four states: the initial state representing the formula, the state representing the temporal subformula $\mathsf{F}^{+}_{<10}b$ and True and False, which indicate termination in an acceptance and a non-acceptance state, respectively. The automaton cycles on the initial state as long as $a \wedge \texttt{evalP(c)}$ is false. When this subformula becomes true, the automaton duplicates itself and a copy keeps on cycling on the initial state, while the other goes in $\mathsf{F}^{+}_{<10}b$ to check whether the consequent is satisfied. This second copy terminates after at most 10 time instants in an acceptance state, if b is true by this deadline, in a non-acceptance state otherwise.

Notice that nothing prevents the system from generating another copy for $\mathsf{F}^{+}_{<10}b$ while one is still active. Indeed, the automaton will produce a copy every time $a \wedge \texttt{evalP(c)}$ is true. Since the copies in $\mathsf{F}^{+}_{<10}b$ will be active at most 10 time units, this means there may be as many as 10 active copies at the same time. This is very costly, but it could be much worse when unbounded operators F^{+} , G^{+} or Until are involved. For instance, with a slightly different formula $\mathsf{G}^{+}((a \wedge \texttt{evalP(c)}) \rightarrow \mathsf{F}^{+}b)$, as before, every time $a \wedge \texttt{evalP(c)}$ is true the automaton generates a copy to check the consequent, but, in this case, $\mathsf{F}^{+}b$ has no constraint on termination, so that infinitely many copies of the automaton may be created. This problem, which seems to seriously affect the feasibility of our approach, can be efficiently overcome without loss of expressive power. In many cases the duplication turns out to be unnecessary, especially in critical systems, where ending in a non-acceptance states is to trigger the agent's exit from the system. Exiting the system at the first failure actually guarantees that the maximum number of needed copies of the ACMA is $C = N + \Sigma_{i=0}^{M} K_i$, where N is the number of temporal subformulae, not including nested X operators, which are as many as M, with K_i being the relevant level of nesting. In other words, the monitoring module just needs one automaton for every temporal operator, except for X^{K_i}, for which K_i copies are required. Let us provide a more detailed account by analyzing each temporal operator.

- $\mathsf{F}^{+}\varphi$: when there is an active automaton waiting for φ to be true, a new copy would be useless, in that it would also wait for the same condition.
- $\mathsf{G}^{+}\varphi$: the existing automaton keeps on being active as long as φ is true, and this is exactly the same task that a new copy would perform.

- $\mathsf{Until}(\varphi, \psi)$: again, a second copy would have the same behavior of the existing one, in that a state with ψ true would satisfy both, a state with φ true and ψ false would keep them both active and waiting for the next instant to perform a new evaluation, and a state in which both propositions are false would lead both copies to a non-acceptance state.
- $\mathsf{X}^K\varphi$: this operator is punctual, in that, the relevant truth value is determined by a single state. Thus, if a new copy of the automaton is required, it is to evaluate a state which lies on the outside of the scope of the currently active automaton. No optimization is then possible. However, each copy will be active for exactly K time instants, which means that no more than K automata will be active at the same time.
- $\mathsf{G}^+_{\bullet K}\varphi$: let us supposed that in the situation depicted in Figure 9, with an automaton launched at instant 1, a new copy is required at instant 2. Automata 1 and 2 aim at checking whether φ is going to be true at all K instants of the intervals starting at 1 and 2, respectively. Resetting automaton 1's counter to zero at instant 2 is a way to achieve the same result without the need for the creation of automaton 2.
- $\mathsf{F}^+_{\bullet K}\varphi$: referring again to Figure 9, if at instant 2 automaton 1 is still active, it means that φ has not become true yet. A new copy created at instant 2 would look for an occurrence of φ in interval $[2, 2 + K]$. It should be noticed that if K does not become true by $1 + K$, automaton 1 ends in a non-accepting state. Thus, the only interval that counts at instant 2 is $[2, 1 + K]$, which means that automaton 1 suffices for the monitoring purposes.

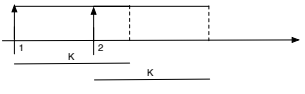


Fig. 9. Timeline to show the multiple copies

Notice that the upper limit proposed for the number of copies of the automaton holds only for critical systems, where the agent is supposed to quit the monitoring process as soon as the first violation occurs. When the level of criticality is not an issue and the monitoring process is performed for statistical purposes, i.e., the monitoring module has to find all the violations and not only the first one, the number of needed copies becomes $C = U + \Sigma_{i=0}^{B} K_i$, where U is the number of unbounded subformulae and B is the number of bounded ones. The idea is that, since now we are interested in all the errors, when a bounded operator is involved, we want to know where the error occurs within the its scope, in order to associate the failure with the relevant event.

4 Related Work

Great interest in monitoring techniques arose in software engineering, and in particular in web services. Among the multitude of works in this field, particularly

related to our work is Dynamo [2], a monitoring framework to assist the execution of workflow processes. This work presents a special purpose monitoring language, called WSCoL (Web Service Constraint Language), used to constrain processes by means of proper monitoring rules, expressed in terms of pre- and post-conditions on the interaction with external partner services. Dynamo adopts a synchronous integration of business and monitoring logics. In fact, the monitoring execution is blocking, i.e., the execution of the business process is suspended every time the monitoring process checks the validity of a rule.

Moreover, in [1] the authors proposed a temporal extension to allow for the monitoring of more complex properties, both functional and non-functional. In this new version the authors define temporal operators exploiting the characteristics of their input language, hence, their semantics is defined on time-stamped words.

Dix et al. [7] propose adding a monitoring agent to a given MAS for debugging purposes. Given a planning problem, the monitoring agent generates all other possible plans to reach the same goal, then continuously checks and compares the messages exchanged by the other agents with all the plans. Should any incompatibility be detected, the monitoring agent generates an error file and reports to the MAS designer.

Guessoum et al. [9] regard monitoring as a process relying on a graph. Each node represents an agent in the MAS, and a weighted arc between two nodes stands for the communication load between the relevant agents. For each node there is a monitoring agent constantly updating the weights of the arcs the node belongs to, and notifying a supervisor agent whenever a significant change in a weight occurs. The supervisor thus has a general view on the communication in the MAS, and may prescribe the replication of some agents to avoid overload.

Kaminka et al. [10] propose a system which, following a non-intrusive approach, bases the monitoring process on overhearing of routine communication between agents which are members of a team aiming at the completion of a specific plan. The team is supposed to be geographically distributed, and the monitoring system sets off inference based on plan recognition against uncertainty due to non-perfect overhearing.

Cranefield [5] presents hyMITL$^{\pm}$, a rule language for specifying social expectations, and outlines an algorithm for rule compliance monitoring. hyMITL$^{\pm}$ relies on a branching model of time and is more expressive than the language we propose, which makes the monitoring process more complex a task. The current status of our work calls for a detailed comparison of the two approaches to establish the correct balance between expressiveness and efficiency. In particular, we hope that our automata-based operational approach will help tackle some of the technical issues in hyMITL$^{\pm}$ related to the evaluation of the formulae's truth value. No implementation detail is explicitly provided in Cranfield's work, but it may be implied that his language is to be used by an inference engine that provides a monitoring service to all subscribing agents.

Cranefield took some further steps along this research path and, together with Winikoff, proposed the model checking of truncated paths as a means to verify social expectations [6]. The authors correctly state that our procedural approach

is not easily comparable to their declarative one, but in our opinion some discussion is still worth because it might shed some light on interesting issues about the design of multi-agent systems. According to the authors, their latest work provides a logical account that their previous efforts were lacking. They tackle two different types of logical issues. First, they deal with the expressiveness of the formal language, which they enrich not only with modal operators dealing with *expectations* (which roughly correspond to our commitments) but also with *nominals* that allow references to the states of the model itself, thus making their logic hybrid. Secondly, as a linear model of time is considered, a problem with contingencies rises: classic truth evaluation would show that a future-directed expectation is fulfilled or violated as soon as such expectation is instantiated. To avoid this type of logical omniscience, the authors rely on a semantics based on truncated paths, according to which sentences must be evaluated only up to specific points of the linear model.

Our approach, on the contrary, is much less ambitious on both accounts. First of all, all deontic aspects are not explicitly modeled in the semantics: commitments are treated as propositions whose truth value changes due to specific actions by the agents in the system; their fulfillment and violation boil down to the truth value of specific sentences. Although lacking a complete formal definition, Cranefield and Winikoff's expectations are modeled in the form of a modal operator Exp. A question thus immediately rises, about whether a modal operator is needed to formalize commitments or expectations. Our point is definitely *not* that Cranefield and Winikoff have endowed their model with an unnecessary burden: as they remark, our approach is operational, so that we need take much less into account. With an aim to a full fledged formal semantics, the use of instruments such as modal operators or truncated path might become inevitable. However, there is another and more interesting question: if a formal logical model is introduced with the final aim to build automata to model check truncated paths, does the bigger effort pay off, if compared to an operational construction of automata that are used to monitor a MAS? The latter solution seems less demanding (i.e. without modal operators, without rather tricky semantical definitions for truncated paths) although guaranteeing the same results.

5 Conclusions and Future Work

In this work we have proposed a monitoring module for analyzing the interaction of a single agent in a MAS. Our monitoring system is based on automata theory and takes advantage of research on model checking and web service monitoring systems. Our monitoring module contains two main components: a Word Composer, devoted to collect data from MAS communication and elaborate them in order to evaluate immediately present and past components, and a Word Analyzer, that analyzes such results to monitor the system during its evolution. There are still some interesting issues to tackle. First, we would like to enrich the logic used to state the properties, in order to define new classes of monitoring problems (e.g.: dense time, data aggregation functions), then, we aim at

investigating the possibility to add a recovery mechanism, so that our monitoring module not only can detect errors, but it may also suggest possible ways to counter them.

References

1. Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S., Spoletini, P.: Validation of web service compositions. IET Software 1(6), 219–232 (2007)
2. Baresi, L., Guinea, S.: Towards Dynamic Monitoring of WS-BPEL Processes. In: ICSOC 2005, pp. 269–282. ACM Press, New York (2005)
3. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
4. Colombetti, M.: A Commitment–Based Approach to Agent Speech Acts and Conversations. In: Agents 2000, Barcelona, Spain, pp. 21–29 (2000)
5. Cranefield, S.: Modelling and monitoring social expectations in multi-agent systems. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS, vol. 4386, pp. 308–321. Springer, Heidelberg (2007)
6. Cranefield, S., Winikoff, M.: Verifying Social Expectations by Model Checking Truncated Paths. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 204–219. Springer, Heidelberg (2008)
7. Dix, J., Eiter, T., Fink, M., Polleres, A., Zhang, Y.: Monitoring Agents using Declarative Planning. In: Günter, A., Kruse, R., Neumann, B. (eds.) KI 2003. LNCS (LNAI), vol. 2821, pp. 646–660. Springer, Heidelberg (2003)
8. Gabbay, D.M.: The declarative past and imperative future: Executable temporal logic for interactive systems. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 409–448. Springer, Heidelberg (1989)
9. Guessoum, Z., Ziane, M., Faci, N.: Monitoring and organizational-level adaptation of multi-agent systems. In: AAMAS 2004, pp. 514–521. ACM Press, New York (2004)
10. Kaminka, G., Pynadath, D., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan recognition approach. JAIR 17, 83–135 (2002)
11. Laroussinie, F., Schnoebelen, P.: A hierarchy of temporal logics with past. Theoretical Computer Science 148(2), 303–324 (1995)
12. Pradella, M., San Pietro, P., Spoletini, P., Morzenti, A.: Practical Model Checking of LTL with Past. In: Proceedings of ATVA 2003 (2003)
13. Ramachandran, D., Amir, E.: Compact propositional encodings of first-order theories. In: Proceedings of AAAI 2005, pp. 340–345. AAAI Press, Menlo Park (2005)
14. Singh, M.P.: Agent Communication Languages: Rethinking the principles. IEEE Computer 31, 40–47 (1998)
15. Spoletini, P.: Verification of Temporal Logic Specification via Model Checking. PhD thesis, Politecnico di Milano, Italy (2005)
16. Thomas, W.: Automata on Infinite Objects. In: Handbook of Theoretical Computer Science. Formal Models and Semantics, vol. B, pp. 133–191 (1990)
17. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Banff Higher Order Workshop, pp. 238–266 (1995)
18. Verdicchio, M., Colombetti, M.: A Logical Model of Social Commitment for Agent Communication. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS, vol. 2922, pp. 128–145. Springer, Heidelberg (2004)

Using SOA Provenance to Implement Norm Enforcement in *e*-Institutions

Javier Vázquez-Salceda and Sergio Alvarez-Napagao

Universitat Politècnica de Catalunya, Spain
{jvazquez,salvarez}@lsi.upc.edu
http://www.lsi.upc.edu

Abstract. In the last 10 years several approaches and technologies other than MAS (such as Web services and Grid computing) have emerged, with the support of the industry, providing their own solutions to distributed computation. As both Web services and Grid computing are based in the concept of service orientation, where all computation is split in independent, decoupled services, there is an opportunity for MAS researchers to test and extend their mechanisms and techniques in these emerging technologies. In this paper we describe a way to adapt the HARMONIA framework to be applied in highly regulated Web services and Grid computing scenarios. To do so we include a *provenance mechanism* as part of our norm enforcement mechanisms, which can be integrated into a SOA Governance workflow. We will show with an example how provenance allows the observation of both service interactions and (optionally) extra information about meaningful events in the system that cannot be observed in the interaction messages.

Keywords: MAS, provenance, web services, SOA, SOA Governance, architecture, electronic institutions, norm enforcement, monitoring.

1 Introduction

With the growth of the Internet and the World Wide Web over the last fifteen years, previous metaphors for computation have been superseded by a new metaphor, of *computation as interaction*, where computing is not an action of a single computer but the result of a network of computers. Multi-Agent Systems (MAS) are one of the technologies that have emerged in this new metaphor. But they are not the only one. In the last 7 years other technologies such as Web services [1] and Grid computing [2] have emerged and matured, with the support of both the research community and the industry. These technologies are based in the concept of service-orientation [3]: a distributed system is comprised of units of service-oriented processing logic (the *services*) which hide their internal logic from the outside world and minimize dependencies among them. Recently some of these service-oriented technologies are converging into a single overarching framework, called Service-Oriented Architectures (SOA). Such framework is creating a collection of best practices principles[1]. But there are still deeper questions in

[1] Some of these principles are service abstraction (beyond what is described in the service contract, services hide logic from the outside world), service loose coupling (services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other) and service autonomy (services have control over the logic they encapsulate). See [3] for more details and patterns in service-oriented design.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 188–203, 2009.

the SOA community regarding the functioning of distributed systems using automated components. Many of these issues have been tackled in the research areas of Artificial Intelligence, Distributed Artificial Intelligence and, in particular, Multi Agent Systems research.

Thanks to the closeness between agent oriented and service-oriented approaches, some cross-fertilization between both technologies is feasible. The SOA community already has identified some potential to integrate agent research in SOA. For instance, Paurobally et. al have proposed to adapt and to refine Multi-Agent Systems research community results to facilitate the dynamic and adaptive negotiation between Semantic Web Services [4]. Foster, Jennings and Kesselman already identified in [5] the opportunity to have some joint research between the Grid and Agents communities.

In our view, there are also opportunities to apply both organizational and institutional approaches in SOA technologies in order to create a social layer on top of existing Web services and Grid platforms. To do so there are two main extensions to be done to SOA platforms:

- The introduction of additional semantics to the communication between services, in order to be able to check the actual behaviour of the actors in a distributed scenario from the intended behaviour.
- The introduction of higher-level behavioral control mechanisms, based in the extraction of some concepts such as commitments, obligations and violations, which can be derived thanks to some intentional stance extracted from the communication semantics.

There have been already some attempts for the first extension. An example is the work presented in [6], where a connection between Agent Communication Languages and Web Service Inter-Communication is proposed, to then extend service communication with some FIPA performatives. The architecture we present in this paper uses this approach.

In the case of the second extension (introducing higher-level behavioral control mechanisms in SOA) it is necessary to have a language and a framework with which to model and manage the commitments. In this paper we present an approach that tackles this issue by defining a provenance-aware norm enforcement framework which combines agents and web services from an institutional approach, using substantive norms and landmarks.

2 SOA behaviour Control and Monitoring

2.1 Provenance

The aim of the IST-funded EU Provenance project was to conceive a computer-based representation of provenance in distributed service-oriented applications that allows users to perform useful analysis and reasoning. The *provenance* of a piece of data is the documentation of the process that produced the data. This documentation can be complete or partial (for instance, when the computation has not terminated yet); it can

be accurate or inaccurate; it can present conflicting or consensual views of the actors involved; it can be detailed or not.

The Provenance architecture assumes that provenance is investigated in open, large-scale systems composed by services, seen as actors, that take inputs and produce outputs. In this abstract view, interactions between actors take place using messages. Actors may have internal states that change during the course of execution. An actor's state is not directly observable by other actors; to be seen by another actor, the state (or part of it) has to be communicated within a message sent by its owner actor. This architecture has formal foundations in the π-calculus [7] and asynchronous distributed systems [8]. The π-calculus is of interest in this context because of its approach to defining events that are internal to actors as hidden communications. This view also allows to formally define mappings with a) Grid applications, b) Web Services and c) Agent-Mediated Services and Applications.

Elements of the Provenance Architecture. The provenance of a data item is represented in a computer system by a set of *p-assertions* made by the actors involved in the process that created it. A p-assertion is a specific piece of information documenting some step of the process made by an actor and pertains to the process. There are three kinds of p-assertions that capture an explicit description of the flow of data in a process. An *interaction p-assertion* is an assertion of the contents of a message by an actor that has sent or received that message. A *relationship p-assertion* is an assertion about an interaction, made by an actor that describes how the actor obtained output data or the whole message sent in that interaction by applying some function to input data or messages from other interactions. An *actor state p-assertion* is an assertion made by an actor about its internal state in the context of a specific interaction.

The long-term facility for storing the provenance representation of data items is the *provenance store*. The provenance store is used to manage and provide controlled access to the provenance representation of a specific data element.

Provenance Life-Cycle. The *provenance life-cycle* is composed of four different phases. First, actors create p-assertions represent their involvement in a computation. After their creation, p-assertions are stored in a provenance store, with the intent they can be used to reconstitute the provenance of some data. After a data item has been computed, users or applications can query the provenance store. At the most basic level, the result of the query is the set of p-assertions pertaining to the process that produced the data. More advanced query facilities may return a representation derived from p-assertions that is of interest to the user. Finally the provenance store and its contents can be managed through a specific interface (subscription management, content relocation, etc).

Provenance Awareness. By transforming a MAS into a provenance-aware MAS, the resulting system gets the capability to produce at execution-time an explicit representation of the distributed processes that take place. Such representation can be then queried and analyzed in order to extract valuable information to validate, e.g.,

the basis of decisions taken in a given case, or to make an audit of the system over a period of time.

2.2 SOA Governance

SOA Governance[2] is an emergent concept in the SOA community used for activities related to exercising control over services [9]. It is a form of electronic governance that has its focus on distributed services and composite architectures, more concretely on SOA scenarios.

In the last years many companies have started to switch to Service-Oriented Architectures for flexibility reasons and to adapt to technologies and practices under continuous growth and standardization. After adopting services as a kind of business asset, SOA Governance has appeared in the form of a methodology which affects the full lifecycle of the services in terms of specification, design, implementation, deployment, management, control, monitoring, maintenance, intercommunication, and redesign. Its aim is to give guidelines on how to establish shared policies, processes, architecture and policies across each layer of an organization.

SOA Governance tries to solve several issues, including: uncontrolled development of services that adapt usual process, usually leading to fragile services less robust than the previous implementation counterparts; lack of reusability, either because they are not designed with reusability in mind, or because they are not seen as valuable components in themselves; security compromise; and unexpected performance.

In summary, SOA Governance is intended to give the methodology and the tools needed to maintain the order in SOA environments. Some reports already try to identify how the community is doing at heading in this direction and which companies are on the good track and what do they lack of [10,11].

There are three steps that define SOA Governance management [9]. Design-Time Governance deals with the definition and application of policies that will govern the design and implementation of Web services in the organization, prior to their deployment in the actual business environment. During Run-Time Governance, policies are defined and enforced in order to govern the deployment, execution, and use of the Web services. Eventually, web services are supposed to be redesigned and reimplemented in order to adapt to business evolving requirements. Change-Time Governance focuses on how the changes on the services affect the behaviour of a whole SOA environment.

The approach currently used in SOA Governance management is based on adding additional Web services in the SOA environment. The main components are:

- Registry: a central catalog for business services.
- Repository: a database of governance policies and metadata.
- Policy enforcement points: services responsible for the enactment of the policies.
- Rules engine: automatic system that manages the enforcement of the policies.
- Configuration environment: user interface for the configuration and definition of policies and governance workflows.

[2] *SOA Governance* should not be confused with *E-Governance*. E-Governance can be defined as the use of Information and Communication Technology as a means to improve transparency, quality and efficiency of service delivery in the public administration.

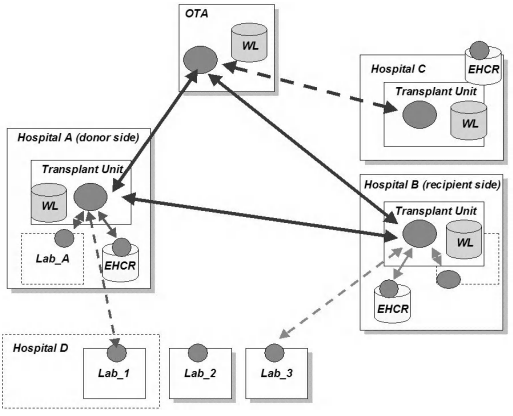


Fig. 1. Actors in the OTMA system. Each medical unit is represented by an agent (circle in figure).

3 Use Case: The Organ Transplant Management Application

The Organ Transplant Management Application (OTMA) is an Agent-Mediated *e*-Institution for the distribution of organs and tissues for transplantation purposes. It extends CARREL [12], the aim of which was to help speeding up the allocation process of solid organs for transplantation to improve graft survival rates. As opposed to CARREL, OTMA uses standard web service technology and is able to interact with provenance stores in order to keep track of the distributed execution of the allocation process for auditing purposes.

Figure 1 summarizes the different administrative domains (solid boxes) and units (dashed boxes) that are modeled in the OTMA system. Each of these interact with each other through agents (circles in the figure) that exchange information and requests through messages. In a transplant management scenario, one or more hospital units may be involved: the hospital transplant unit, one or several units that provide laboratory tests and the Electronic Healthcare Record (EHCR) subsystem which manages the health care records for each institution. The diagram also shows some of the data stores that are involved: apart from the patient records, these include stores for the transplant units and the Organ Transplant Authority (OTA) recipient waiting lists (WL). Hospitals that are the origin of a donation also keep records of the donations performed, while hospitals that are recipients of the donation may include such information in the recipient's patient record. The OTA has also its own records of each donation, stored case by case.

4 A Normative Framework Based in Norms and Landmarks

We use HARMONIA[13] as the basis for our normative framework, although the connection between the ideal states in the norms and the actual execution states of the system is done through the concept of landmarks, as in [14].

```
Norm       OTM:N37
Condition  OBLIGED(hospital
             DO ensure_compatibility(organ, recipient))
             BEFORE (allocator DO assign(organ, recipient)))
Violation  NOT(done(ensure_compatibility(organ, recipient))
condition    AND done(assign(organ, recipient))
Sanction   inform(board, "NOT(done(ensure_compatibility(organ,
             recipient)) AND done(assign(organ, recipient))")
Repairs    {stop_assignation(organ);
            assert(
              NOT(done(ensure_compatibility(organ, recipient))
               BEFORE done(assign(organ, recipient)), p_store
            );
            wait(asserted(
            ensure_compatibility(organ, recipient)));
            resume_assignation(organ); }
```

Fig. 2. Example of an OTMA norm

In our normative framework we propose that enforcement of norms should not be made in terms of direct control of a central authority over the goals or actions that the agents may take, but through the detection of the *violation states* that agents may enter into and the definition of the *sanctions* that are related to the violations. With this approach we do not make strong assumptions about the agents' internal architecture, as the *e*-Institution only monitors the agent behaviour (that is, agents are seen as *black boxes*.). The enforcement of the norms in an *e*-Institution is achieved through a special kind of agents, the *Enforcement Agents*, which monitor the behaviour of the agents, detect violations and check the compliance of the sanctions.

4.1 Norms in Organ Transplant Management

We use a language for substantive norms [15] which is an evolution of the original norm language in HARMON*IA*. Its central element is the norm condition, based in deontic concepts (OBLIGED, PERMITTED, FORBIDDEN) which can be conditional (IF) and can include temporal operators (BEFORE, AFTER). The violation is a formula derived from the norm to express when a violation occurs. The sanction field is a set of actions which should be executed when a violation occurs (e.g. imposing a fine, expulsion of an agent), while the repairs field contains a set of actions to undo the negative effects of the violation. The language also included the specification of the detection mechanism, but in our provenance-based enforcement architecture this is no longer needed.

An example (extracted from organ and tissue allocation regulations) is presented in Figure 2. It expresses the obligation of the hospital to carry on the compatibility tests for a potential recipient of a given organ before assigning the organ to that recipient. The violation condition defines when the violation of that norm occurs. In this scenario, the sanctions field applies an indirect punishment mechanism (reputation) to the hospital, by informing about the incident to the board members of the transplant organization. The repair plan consists of stopping the assignation process, recording the incident in the provenance store (which acts as a log) and then wait for the compatibility test to be performed.

It is important to note that the combination of violation and sanction handling provides a flexible way to implement safety control of a medical system's behaviour (i.e., avoid the system to enter in a undesirable, illegal state because of a failure in one of the agents).

```
Norm        OTM:N37
Violation   NOT(done(ensure_compatibility(organ, recipient)) AND
condition     done(assign(organ, recipient))
Detection   (NOT(
condition       asserted(ensure_compatibility(organ, recipient), t1)
                AND asserted(assign(organ, recipient), t2)
              ) OR
              ((asserted(ensure_compatibility(organ, recipient), t1)
               AND asserted(assign(organ, recipient), t2)
               AND (< t2  t1))
```

Fig. 3. Example of a violation handling rule

4.2 Control Landmarks

Landmarks [16] are often used with similar purposes in order to provide abstract specifications of organizational interaction in general. Landmarks are formalized as state descriptions, which are partially ordered in directed graphs to form landmark structures which are called *landmark patterns*.

In our case we extend the use of landmarks to represent highly relevant positive and negative states of the system (positive and negative landmarks) and the partial ordering between those states imposed by the regulations or practices. For instance, in the norm in Figure 2 we can identify two critical states as landmarks, the one where `ensure_compatibility` happens and the one where `assign` happens. The norm also imposes a partial ordering where the former should always happen before the latter.

Given the set of landmark patterns coming from the institution, agents may reason about the exact sequencing of actions or the protocol to use to pass from one landmark state to the other. This even allows an agent to create acceptable variations of a predefined protocol that are legal and that allow them to fulfill their interests or to cope with an unexpected situation not foreseen in the protocol. Given some landmarks, agents may even negotiate the protocol to use.

Landmarks can be used as checkpoints by the enforcing agents (e.g. whenever the assignation is done, it should be the case that previously the compatibility check was done). In short, norm enforcement can be done by checking that the system as a whole passes only through positive landmarks during its execution and in the proper order. In our system, landmarks are mapped into conjunctions of p-assertions, and landmark ordering is expressed in rules by means of the time stamps attached to each p-assertion. Figure 3 shows an example of how these p-assertions can be then used to detect a violation of the norm in Figure 2.

5 An Architecture Proposal for Norm Enforcement in *e*-Institutions Based in Provenance

In this section we introduce our proposal for a generic Provenance-based norm enforcement architecture. Although current version is mainly designed for Web service and Grid platforms, it can be easily adapted to be used also by agents in an agent platform. The global picture of this architecture is shown in Figure 4. When application agents enter for the first time in the *e*-institution, they can access the norms, the ontological

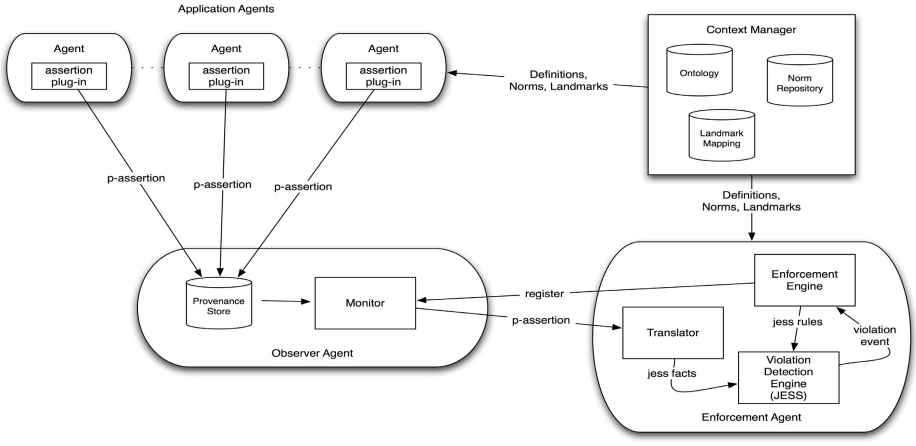


Fig. 4. A generic Provenance-based norm enforcement architecture

definitions and the landmark definitions in the context manager module. Agents log the relevant events by creating p-assertions that are sent to the observer agent, which is the one that keeps the Provenance store that acts as a log for all the reported events. The observer agent sends some of those reported p-assertions to one or more Enforcement agents (each of those should have previously registered the list of p-assertions they need to be notified, according to the norms each of them has to enforce). Each enforcement agent combining the reported events in the p-assertions with the norms and landmarks that such agent is responsible to enforce. If a violation is detected, then the enforcement agent should execute the sanction and repair plans, as specified in the norms.

It should be noted that the *e*-Institution framework, HARMON*IA*, does not need to be modified for using Provenance. Provenance is only a different way to observe the state of a distributed system, which records and provides inputs (events and actions) that can be used for norm enforcement.

The following sections describe in detail each of the actors in our proposed architecture, focusing on their main roles and components.

5.1 Context Manager

In the approach taken for the architecture, every *e*-institution defines a normative context. This context gathers all the elements needed for understandability and interoperation between the agents belonging to a specific institution. The Context Manager is a registry responsible for the management of these elements and for providing to the agents any information related to the normative context.

An instance of this registry will represent a specific normative context, and will contain:

- a specific vocabulary defining the meaning of the terms used in the interactions between the agents of the institution,
- shared descriptions about processes and actions in the domain, and
- the norms that may affect the interactions between parties bound to the context.

To fulfill its responsibilities, the Context Manager has three main components, explained in the next subsections.

Ontology. The Ontology is a repository which stores definitions of terms, as well as references to definitions, for the data models of the context. This ontology should define, for a given domain, terms such as objects and entities (e.g. patient, doctor, organ, kidney), predicates (e.g. compatible(organ, recipient)) and actions (e.g. assign(organ, recipient)). In our architecture the ontology plays an important role as it should fix the interpretation for all terms that appear in the norms to be enforced.

Norm Repository. This module is responsible for storing and managing the norms of the *e*-institution. Each norm includes not only the deontic expression but also the violation condition, the sanction plan and the repairs plan.

Landmark Mapping. This module is responsible for storing the mapping between landmarks and p-assertions. Such mappings can be used by both a) the application agents, to use the same p-assertion structure when reporting a relevant event that is listed as a landmark in the normative context of the *e*-institution; and b) the enforcement agents, that can use these mappings to translate the p-assertions they receive from the observer agent into landmarks.

5.2 Application Agent

The Application Agents are those agents that interact within each other inside the *e*-institution and its context. They have the same generic role as the agents in any typical multi-agent system and they do not necessarily have an active role in norm enforcement, but they should report all relevant events to the observer agent by creating p-assertions, which will be used by the enforcement agents to enforce the norms applying to the application agents' behaviour. P-assertion creation and reporting is handled by the p-assertion plug-in, a middleware component common to all Application Agents.

Before an Application Agent can start its activity within the *e*-institution, it has to retrieve the definitions, norms and landmarks of the context from the Context Manager. In this paper we make no assumption about the internal architecture of the agent and how this knowledge can be incorporated in the agent reasoning cycle. We also make no assumption about the exact technological platform in which it is implemented: it can be either a Web service, a Grid service or even a FIPA-compliant agent with a service wrapper that allows the agent to interact with the other actors in the architecture. Our only assumption is that the agents internal reasoning cycle has been modified to be able to report meaningful events (landmarks) through the Assertion Plug-in.

Assertion Plug-In. This component is a middleware plug-in which manages the interaction between the application agents and the Provenance Store, ensuring a safe, reliable, and accurate recording of the events and landmarks generated by the agents execution. Whenever an agent wants to report the occurrence of a landmark:

1. The Assertion Plug-in translates this landmark into one or more p-assertions, by following the landmark mapping rules retrieved from the Context Manager.
2. The Assertion Plug-in sends the p-assertion(s) to the Observer Agent by using the Provenance Client API.

To avoid that p-assertions stop the execution of the agent or that some p-assertions get lost due to temporary unavailability communication problems between the Application Agent and the Observer Agent, the plug-in uses a p-assertion queue, which allows the p-assertion submission to be completely asynchronous and loosely coupled to the core of the agent, avoiding critically blocks in its execution.

5.3 Observer Agent

An Observer Agent has the responsibility to safely register and maintain the environmental events and state changes of the *e*-institution. The information gathered is then used in the norm enforcement, by providing selected pieces of information to the interested Enforcement Agents.

The gathering and the selection are critical processes. Some possible errors which depend on the Observer Agent and could compromise norm enforcement can take place, for example, if the events logged are not complete or reliable enough, or if the information provided to the Enforcement Agents doesn't match with their needs or arrives too late.

The gathering is handled by the Provenance Store which, along with the Assertion Plug-in, offers the proper recording functionalities. The Monitor acts as a link between this repository and the Enforcement Agents, offering registering and notification mechanisms. Both Observer Agent components are described in the subsections below.

Monitor. The Provenance Store works only in a *push* way. The Enforcement Agents preferably need a real-time accurate representation of the *e*-institution, so the Observer Agent, as an actor, should behave in a *pull* way. That is why we have implemented the Monitor, layered on top of the Provenance Store. This component will keep an accurate real-time representation of the p-assertions being recorded in the Provenance Store.

Of course, this job should be handled efficiently, not only in time, but also in space, only keeping pointers to the p-assertions that are for some interest for the other agents. A registry is therefore incorporated to the Monitor, to which the Enforcement Agents subscribe with a list of mapped landmark patterns. While continuously reconstructing the real-time *picture* of the *e*-institution, the Monitor will just query those p-assertions which match with the patterns of the Enforcement Agents registered. As soon as a p-assertion has appeared in the Provenance Store that matches a registration pattern of an Enforcement Agent, this p-assertion is sent to the registrant.

Provenance Store. The Provenance Store is usually an independent service, but we consider it as part of the Observer Agent, as these will be the only actors of the *e*-institution which will make use of them. As a repository of raw p-assertions, it will only receive one kind of input, provided by the Assertion Plug-ins of the Application Agents. As well, it will only generate one kind of output, in this case the result of the queries made by the Monitor, as sets of p-assertions.

5.4 Enforcement Agent

The Enforcement Agents are responsible for the fulfillment of a subset of the norms of the context in the *e*-institution. This requires them to have a complete knowledge of the

context, by retrieving the descriptions and the norms from the Context Manager, as well as a complete knowledge of all the events in the system related to the norms they have to enforce. These enforcement is then guaranteed by a) firstly detecting the violations, and then b) applying the corresponding sanctions.

In order to generate the knowledge about the events, these agents take profit of the Observer Agent by registering the set of landmarks they are supposed to look after. Once registered, they will be properly notified in the form of p-assertions. Therefore, there is no need of a direct communication between an Enforcement Agent and the Application Agents. The Translator converts these p-assertions into a format understandable by the Enforcement Agent. Another component is needed for detecting the violations. In our case we are using a Jess engine, which matches the events, in the form of Jess facts, and the norms, in the forms of Jess rules. The Enforcement Engine is responsible for registering to the Observer Agents and applying sanctions. A further explanation of how this component works is also included below.

Translator. The Observer Agent sends p-assertions to the Enforcement Agent when they are of any interest. However, the Violation Detection Engine is an instance of a Jess engine. The Translator is a simple component which parses these p-assertions and generates Jess facts.

The Translator obtains the translation rules from the Context Manager. In Figure 5 we show one example of a rule that obtains a Jess assertion of an organ assignment, taking an organ assignment p-assertion as input. This rule parses the XML formatted p-assertion, keeping only the relevant data for the system and generating an asserted fact, which will be added to the Jess engine. In this case, the rule is involved in the moment that the doctor of a hospital accepts the organ offer and therefore confirms the assignment proposed by the OTA. According to the medical protocol being followed, the relevant pieces of data in this step are the exact moment of the assignment, the recipient patient identifier, and the organ. They are retrieved from the XML p-assertion and written in a Jess fact.

```
(defrule OTM-RULES-MODULE::assertconfirmassignment
  (MAIN::Element (LocalName "opencontent")
    (ElementID ?content))
  (MAIN::Element (LocalName "timestamp")(Text ?timestamp)
    (ParentID ?content))
  (MAIN::Element (LocalName "confirmassignment")
    (ElementID ?confirmassignment)(ParentID ?content))
  (MAIN::Element (LocalName "organ")(Text ?organ)
    (ParentID ?confirmassignment))
  (MAIN::Element (LocalName "pid")(Text ?pid)
    (ParentID ?confirmassignment))
  (not (OTM-RULES-MODULE::confirmassignment
    (ElementID ?confirmassignment)(timestamp ?timestamp)
    (organ ?organ)(pid ?pid)))
=>
  (assert (OTM-RULES-MODULE::confirmassignment
    (ElementID ?confirmassignment) (timestamp ?timestamp)
    (organ ?organ)(pid ?pid))))
```

Fig. 5. An example of translation rule from p-assertion to Jess $asserted$ fact

When an agent records a p-assertion indicating the confirmation of an assignment, it includes content compliant with the OTMA XML schema. On the left side, this rule matches one by one the elements contained inside the *opencontent* element: the exact moment of the action, the name of the event (*confirmAssignment*), and inside the *confirmAssignment* element, the organ being proposed for reception and the ID of the recipient. After the matching, the left side of the rule checks that there was no assertion made yet for the same event. On the right side, the rule asserts the event *confirmAssignment* into the base of facts.

He have implemented an automatic translator of rules, capable of parsing an schema and generating one rule per each kind of event the content of the p-assertion might contain, which right now we assume is once per each XML element defined. It will be improved in future releases.

Violation Detection Engine. Once the Enforcement Engine has received the norms from the Context Manager, it creates a set of Jess rules out of them and sends them to the Violation Detection Engine. This component is, in fact, an instance of a Jess engine which will execute these rules with the facts provided by the Translator. Whenever a violation is detected, the Enforcement Engine is conveniently informed.

Enforcement Engine. The Enforcement Engine is the component of the Enforcement Agent that takes decisions and plans actions whenever a violation is raised. In order to interact with the Violation Detection Engine, this component needs to provide Jess rules for each norm.

The violation for the norm *N37* has to be raised whenever, in the confirmation of an assignment, this assignment has been made before having checked for compatibility. This might happen when the assignment is done but the compatibility is never ensured. But also when both things are done, but in the wrong order. This second case is the one depicted in Figure 6. The rule shown in the figure takes as input two facts: the fact generated (using the translation rule shown in Figure 5) when the hospital confirmed the assignment of the offered organ to the doctor, and the fact generated when the organ was tested for compatibility. The third condition of the rule, `(< t2 t1)`, will become true if the assignment has been done before the compatibility test. Whenever the rule gets executed, a violation fact for the norm *N37* will be added to the Jess engine and the Enforcement Agent will, at some point, take measures to repair the violation.

The Enforcement Agent will act accordingly to the type of measures needed. If the sanction or the repair measures require that a specific Application Agent executes a certain action, that agent will be informed of that. On the other hand, the sanction or the

```
(defrule OTM-RULES-MODULE::eventOTM_N37_2
(OTM-RULES-MODULE::ensure_compatibility (organ ?organ)
  (recipientID ?recipientID)(timestamp ?t1))
(OTM-RULES-MODULE::assign (organ ?organ)
  (recipientID ?recipientID)(timestamp ?t2))
(< t2 t1)
=>
(assert (OTM-RULES-MODULE::violation (norm OTM_N37)
  (organ ?organ)(recipientID ?recipientID)))
```

Fig. 6. An example of violation detection rule in Jess

repair measures that involve the institution as itself will be carried into effect by the Enforcement Agent. When an Enforcement Agent is initiated, the ontological definitions and the norms of the context are stored in its Enforcement Engine. This component is also the responsible for registering to the Monitor.

For the norm example shown in Figure 2, all the measures should be executed by the Enforcement Agents, as they are all institutional.

6 Discussion

The use of norms as a mechanism to model the expected (proper) behaviour in a computational system has led to the logical related issue of their enforcement. Several aspects of norm enforcement have been already studied in theory: some approaches use sanctions imposed and enforced by specially empowered agents [17,18,19,13,15,20] to reduce the likelihood that agents choose to perform unwanted behaviour, while others use of some shared normative reputation mechanism [21,22,23] that reduces future interaction with agents not behaving properly. In the former norm enforcement is usually centered in special agents [13] or platform mechanisms [20] responsible for behaviour monitoring to detect violations to the norms; these approaches put few requirements on the rest of the agents in the system but require a high level of trust in the enforcer agents. In the latter norm enforcement is usually distributed in all the agents in the system, which collectively ostracise agents not abiding to the norms [23]. There are also some distributed sanction-based approaches [18] where the monitorisation and enforcement are distributed in all agents. While in some setups distributed enforcement may be effective (specially in society with simple normative models that are easy to compute), these also reduce the openness of thew resulting system, as they may impose strong assumptions on all the agents entering into the system (all them should understand the norms and the norm enforcement mechanisms and be able to implement it). In this work we have chosen to allocate behaviour monitoring and norm enforcement in specially designed, trusted components and agents in order to reduce the requirements in other agents' and services' internal architecture and reasoning capabilities.

Although there are lots of theoretical research on norm enforcement, there is far less work on implementations of enforcement mechanisms that can be applied to substantive norms in highly regulated environments. AMELI [24] is a toolkit for the specification and verification of agent mediated *e*-institutions that based on a dialogical framework. In this framework, all observable activities from the agents are seen as messages in the context of a *scene*. In AMELI all norms are regimented through the specification of a pre-defined protocol, guaranteeing norm-compliance of agents by restricting the set of possible actions to the ones defined in the protocol.

In [15,16] there is a first exploration of substantive norms already applied to AMELI. The main difference in our approach is that we can also include internal information from agents which is not part of any interactions. On the other hand, the formalism defined in [25] only considers messages as observable events.

[26] introduces integration of *e*-Institutions in Grid environments by extending the GRIA framework, which is based on basic web services. Our solution gives more flexibility to the behaviour of the services, as norms are substantive and not rigidly regulated.

It is important to note that the provenance mechanism used here is an implementation of an open architecture [27] that ensures interoperability in heterogeneous systems without compromising security or scalability. Other provenance mechanisms are mainly based on a middleware layers which only capture interactions. This kind of provenance mechanisms can be used in less regulated environments but bring little extra power to the existing mechanisms in agent-mediated *e*-Institutions.

Academic research on SOA Governance is still not abundant, but there are already some interesting proposals of models [28], methodologies [29,30] and frameworks [31]. In our paper we present a novel approach to the topic based on flexible normative enforcement via landmark monitoring.

7 Conclusions

The fact that the SOA business community is concerned about how to define and manage policies for the definition, deployment and change of Web services is a clear sign that organizations need to translate and adapt their own business processes and methodologies of work in their SOA environments. Electronic Institutions respond to the need of regulation in MAS that have to be bound to certain norms that apply in the context of an institution. They provide a theoretical solution that could match many of the needs of SOA Governance as, once the policies are defined, an *e*-Institution framework could take care of their enforcement.

However, SOA Governance does not focus on MAS. With our architecture proposal we aim at bridging this gap by combining Web services and agents inside a normative framework derived from HARMON*IA*, and deployed in heterogeneous (MAS, Web services and/or Grid) platforms.

As next steps we will define a mapping between the operational representation of norms and: 1) orchestration languages, which would allow us to better integrate our proposed architecture into business processes, as well as 2) choreography languages, which would give us the possibility of extending the uses of the interaction provenance recording. By defining a mapping, norms could be instantiated in languages like WS-BPEL or WS-CDL, which could be imported directly by web service workflow engines which are currently widely used.

Acknowledgment

This work has been funded by IST-2002-511085 PROVENANCE and IST-2006-034418 CONTRACT projects. Javier Vázquez-Salceda's work has been also partially funded by the "Ramón y Cajal" program of the Spanish Ministry of Education and Science.

References

1. World Wide Web Consortium (W3C): Web services architecture (February 2004), `http://www.w3.org/TR/ws-arch/`
2. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco (1998)

3. Erl, T.: Service-Oriented Architecture. Prentice Hall PTR, NJ (2004)
4. Paurobally, S., Tamma, V., Wooldridge, M.: Cooperation and agreement between semantic web services. In: W3C WS on Frameworks for Semantics in Web Services (January 2005)
5. Foster, I., Jennings, N.R., Kesselman, C.: Brain meets brawn: why Grid and agents need each other. In: 3rd Int. Conf. on Autonomous Agents and MAS, pp. 8–15 (2004)
6. Willmott, S., Pea, F.O.F., Merida-Campos, C., Constantinescu, I., Dale, J., Cabanillas, D.: Adapting agent communication languages for semantic web service inter-communication. In: WI 2005: Proc. of the 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence, pp. 405–408. IEEE Computer Society, Washington (2005)
7. Milner, R.: Communicating and Mobile Systems: the Pi-Calculus. Cambridge University Press, Cambridge (1999)
8. Lynch, N.: Distributed Algorithms. The Morgan Kaufmann Series in Data Management Systems (January 1996)
9. webMethods: SOA Governance: Enabling Sustainable Success with SOA (2006), http://www1.webmethods.com/PDF/whitepapers/
10. Fulton, L.: The Forrester Wave: SOA Service Life-Cycle Management. In: Q1 2008 (March 2008)
11. Kenney, L.F., Plummer, D.C.: Magic Quadrant for Integrated SOA Governance Technology Sets, 2007. Gartner RAS Core Research Note G00153858 (March 2008)
12. Vázquez-Salceda, J., Cortés, U., Padget, J., López-Navidad, A., Caballero, F.: The organ allocation process: a natural extension of the carrel agent-mediated electronic institution. AI Commun. 16(3), 153–165 (2003)
13. Vázquez-Salceda, J.: The role of norms and electronic institutions in multi-agent systems. The HARMONIA framework. PhD Thesis. Whitestein Series in Software Agent Technologies. Birkhäuser (2004)
14. Aldewereld, H., Grossi, D., Vázquez-Salceda, J., Dignum, F.: Designing Normative Behaviour via Landmarks. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM 2005 and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 157–169. Springer, Heidelberg (2006)
15. Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Operationalisation of norms for usage in electronic institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 163–176. Springer, Heidelberg (2007)
16. Aldewereld, H.: Autonomy vs. conformity: an institutional perspective on norms and protocols. PhD Thesis (2007)
17. Boella, G., van der Torre, L.: Enforceable social laws. In: AAMAS 2005: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 682–689 (2005)
18. Heckathorn, D.: Collective sanctions and compliance norms: A formal theory of group-mediated social control. American Sociological Review 55(3), 366–384 (1990)
19. Lopez y Lopez, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: AAMAS 2002: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems. ACM Press, New York (2002)
20. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.P.M.: Implementing Norms in Multiagent Systems. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 313–327. Springer, Heidelberg (2004)
21. Castelfranchi, C., Conte, R., Paoluccci, M.: Normative reputation and the costs of compliance. Journal of Artificial Societies and Social Simulation 1(3) (1998)

22. Grizard, A., Vercouter, L., Stratulat, T., Muller, G.: A peer-to-peer normative system to achieve social order. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 274–289. Springer, Heidelberg (2007)
23. Perreau de Pinnick, A., Sierra, C., Scholermer, M.: Distributed norm enforcement via ostracism. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 301–315. Springer, Heidelberg (2008)
24. Esteva, M.: Electronic Institutions: from specification to development. PhD Thesis, UPC (April 2003)
25. Cliffe, O., De Vos, M., Padget, J.: Embedding Landmarks and Scenes in a Computational Model of Institutions. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 41–57. Springer, Heidelberg (2008)
26. Ashri, R., Payne, T.R., Luck, M., Surridge, M., Sierra, C., Aguilar, J.A.R., Noriega, P.: Using Electronic Institutions to secure Grid environments. In: Klusch, M., Rovatsos, M., Payne, T.R. (eds.) CIA 2006. LNCS, vol. 4149, pp. 461–475. Springer, Heidelberg (2006)
27. Groth, P., Jiang, S., Miles, S., Munroe, S., Tan, V., Tsasakou, S., Moreau, L.: An architecture for provenance systems (February 2006)
28. Derler, P., Weinreich, R.: Models and Tools for SOA governance (2007)
29. Zhou, Y.C., Liu, X.P., Kahan, E., Wang, X.N., Xue, L., Zhou, K.X.: Context aware service policy orchestration. ICWS, 936–943 (2007)
30. Papazoglou, M.P.: Service-oriented design and development methodology. Int. Journal of Web Engineering and Technology 2, 412–442 (2006)
31. Kajko-Mattsson, M., Lewis, G.A., Smith, D.B.: A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems. In: ICSEW 2007: Proc. of the 29th Int. Conf. on Software Engineering WS, p. 117. IEEE Computer Society, Washington (2007)

Verifying Social Expectations by Model Checking Truncated Paths

Stephen Cranefield[1] and Michael Winikoff[2,3⋆]

[1] Department of Information Science
University of Otago, Dunedin, New Zealand
scranefield@infoscience.otago.ac.nz
[2] Higher Education Development Centre
University of Otago, Dunedin, New Zealand
michael.winikoff@otago.ac.nz
[3] School of Computer Science and Information Technology
RMIT University, Melbourne, Australia

Abstract. One approach to moderating the expected behaviour of agents in open societies is the use of explicit languages for defining norms, conditional commitments and/or social expectations, together with infrastructure supporting conformance checking. This paper presents a logical account of the fulfilment and violation of social expectations modelled as conditional rules over a hybrid linear propositional temporal logic. Our semantics captures the intuition that the fulfilment or violation of an expectation must be determined without recourse to information from later states. We define a means of updating expectations from one state to the next based on formula progression, and show how conformance checking was implemented by extending the MCLITE and MCFULL algorithms of the Hybrid Logics Model Checker.

1 Introduction

An *electronic institution* [1] is an explicit model of the rules, or norms, that govern the operation of an open multi agent system. A given electronic institution provides rules that agents participating in the institution are expected to follow. These rules can include more traditional protocols (e.g. a request message comes first, followed by either an accept or a refuse), as well as properties that are expected to apply to complete interactions, for example, the norm that any accepted request must be eventually fulfilled.

Since electronic institutions are open systems it is not possible to assume any control over agents, nor is it reasonable to assume that all agents will follow the rules applying to an interaction. Instead, the behaviour of participating agents needs to be monitored and checked, with violations being detected and responded to in a suitable way, such as "punishing" the agent by applying sanctions, or reducing the agent's reputation.

There is therefore a need for mechanisms to check for the fulfilment or violation of norms with respect to a (possibly partial) execution trace. Furthermore, such a mechanism can also be useful for rules of social interaction that are less authoritative than centrally established norms, e.g. conditional rules of expectation that an agent has

⋆ Since writing this paper Winikoff has left RMIT.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 204–219, 2009.

established as its personal norms, or rules expressing learned regularities in the patterns of other agents' behaviour.

Thus in this paper we focus on modelling the general concept of *social expectation* and demonstrate the use of *model checking* for detecting the fulfilment or violation of such expectations by extending the MCLITE and MCFULL algorithms of the Hybrid Logics Model Checker [2]. The advantages of building on model checking, rather than implementing our own checking algorithm (as was done previously [3]) are that we are working within a clearly defined and well studied verification framework, and that it allows us to extend existing software including a range of optimisations that have been developed for model checking. Although the problem of model checking, in its full generality, is more complex than we need, the problem of model checking a path (a finite or ultimately periodic sequence of states) has also been studied and "can usually be solved efficiently, and profit from specialized algorithms" [4]. We have therefore investigated the applicability of model checking as a way of checking for expectations, fulfilments and violations over a model that is a linear history of observed states.

The theory underlying our approach is designed to apply equally well to both *online* and *offline* monitoring of expectations, a distinction that has not been made in previous work. For online monitoring, each state is added to the end of the history as it occurs, and the monitoring algorithm works incrementally. The underlying formalism can assume that expectations are always considered at the last state in the history. In contrast, in the offline mode, expectations in previous states are also checked. At each past state, the then-active expectations must be checked for fulfilment without recourse to information from later states: the truth of a future-oriented temporal proposition ϕ at state s over the full history does not imply the fulfilment at s of an expectation with content ϕ.

This paper is structured as follows. Section 2 outlines our intuitions about expectations, fulfilment and violation and sketches out our logical account of these concepts. Section 3 describes the logic and semantic mechanisms we use to express fulfilment and violation of an expectation. In Section 4 we give a brief description of formula progression, a technique used to express the evolution of an unfulfilled and non-violated expectation from one state to the next. Section 5 then describes the Hybrid Logics Model Checker that we have used in this work and the extensions we have made to it. Example output from the extended model checker for two example scenarios is presented in Section 6. Finally we discuss related work in Section 7 and summarise the paper and plans for future work in Section 8.

2 Formalising Expectations, Fulfilment and Violation

In this work we study the general notion of *expectations*. It is our position that the base-level semantics of expectations with different degrees of force (expectations inferred from experience, promises, formal commitments, etc.) are the same. The differences between these lie in the pragmatics of how they are created and propagated, how their fulfilment and violation is handled, and the type of contextual information associated with them (e.g. the debtor and creditors associated with a commitment).

Our intuition behind expectations is that they are created in some context which may depend on the current and recorded past states of an agent (including any representation

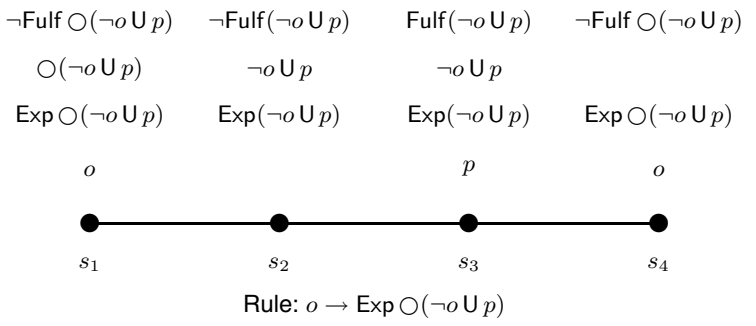


Fig. 1. An example rule and scenario

it has of the external environment), and that the created expectation is a constraint indicating the expected future sequences of states. We model this by conditional rules:

$$\lambda \rightarrow \mathsf{Exp}\,\rho$$

where λ and ρ are linear temporal logic expressions with λ referring to the past and present and ρ encoding the constraint on the future. The modality Exp is needed as it is not guaranteed that ρ will hold; it will just be "expected" if the condition holds.

The question then arises of when an expectation should be considered to be fulfilled (denoted $\mathsf{Fulf}(\phi)$) or violated ($\mathsf{Viol}(\phi)$). Consider Fig. 1. This shows a rule expressing an expectation on the interaction between a merchant and a customer: a customer agent that has placed an order (modelled as the proposition o) should not subsequently place another order until its order has been paid for (proposition p). We formalise this as $o \rightarrow \mathsf{Exp}\bigcirc(\neg o\,\mathsf{U}\,p)$, i.e., when o holds, it is expected that, from the next state on, o is false until p holds. In the figure, the bottom row of formulae show the propositions that are observed in a segment of one possible history: o holds in states s_1 and s_4. and p holds in s_3. The row above this shows the expectations that are *created* by the rule (in states s_1 and s_4) and then *updated* from one state to the next (in states s_2 and s_3, using a technique discussed below). Above this we show the content formulae of these expectations for the first three states, which can easily be seen to hold due to the semantics of the temporal operators $\bigcirc$ and U (a longer segment is needed to evaluate the content of the expectation in s_4). However, as indicated in the top row, we should not necessarily conclude that these expectations are fulfilled just because their content formulae are true. The determination of fulfilment and violation must be made *without recourse to future information*. Thus, only in state s_3, when payment is made (p holds), should it be concluded that the current expectation is fulfilled. Section 3 presents a temporal operator $\mathsf{Trunc_S}$ that allows us to express this restriction to past and present information. For now, we will assume that we have suitable definitions of $\mathsf{Fulf}(\phi)$ and $\mathsf{Viol}(\phi)$, and move on to consider how expectations evolve from one state to the next.

We assume that an expectation can be fulfilled or violated at most once and that an expectation that is not fulfilled or violated in a state should persist (in a possibly modified form) to the next state:

$$\mathsf{Exp}\,\phi \wedge \neg\,\mathsf{Fulf}\,\phi \wedge \neg\,\mathsf{Viol}\,\phi \rightarrow \bigcirc\,\mathsf{Exp}\,\psi$$

What should ψ be? Although at least one alternative approach exists (see Section 7), we believe that the most intuitive representation of an expectation is for it to be expressed in terms of the current state. Thus ψ should represent a change of viewpoint of the constraint represented by the expectation ϕ from the current state to the next state. This can be seen in Fig. 1 where $\mathsf{Exp}\bigcirc(\neg o \,\mathsf{U}\, p)$ from s_1 becomes $\mathsf{Exp}(\neg o \,\mathsf{U}\, p)$ in s_2 and then remains as $\mathsf{Exp}(\neg o \,\mathsf{U}\, p)$ in s_3 as p was not true in s_2.

The transformation of ϕ into ψ should also take into account any simplification of the expectation due to subformulae of ϕ that were true in the current state. Thus, an expectation $\mathsf{Exp}(p \wedge \bigcirc q)$ should become $\mathsf{Exp}\, q$ in the next state if p holds currently. This is precisely the notion of formula progression through a state [5]. Formula progression (which will be explained in more detail in Section 4) allows us to complete our informal characterisation of the evolution of expectations through time:

$$\mathsf{Exp}\,\phi \wedge \neg\,\mathsf{Fulf}\,\phi \wedge \neg\,\mathsf{Viol}\,\phi \wedge \mathsf{Progress}(\phi, \psi) \rightarrow \bigcirc\,\mathsf{Exp}\,\psi$$

This conception of expectation, fulfilment and violation has been implemented in a previous progression-based system using a logic combining future and past temporal operators with the guarded fragment of first order logic, binders and a form of nominal [3]. The logic allows the expression of temporally rich conditional expectations such as "Once payment is made, the service-providing agent is committed to sending a report to the customer once a week for 52 weeks or until the customer cancels the order". However, although this system used a logical notation for rules of expectation, the detection of fulfilments and violations and the progression of expectations from one state to the next were handled algorithmically, and there was not a logical account of these notions. This paper provides such a logical account, elaborating on the intuition presented above, and demonstrates how to build semantics corresponding to this intuition into a model checker for detecting expectations and their fulfilment and violation.

3 Formal Background

The logic we use to model social expectations is a hybrid temporal logic that is an extension of the one implemented by the Hybrid Logics Model Checker [2]. It is described by the following grammar:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \ominus\phi \mid \phi_1 \,\mathsf{U}\, \phi_2 \mid \phi_1 \,\mathsf{S}\, \phi_2 \mid x \mid n \mid @_t\phi \mid \downarrow x\phi \mid \mathsf{E}\,\phi$$

where p is a proposition, $\bigcirc$ is the standard temporal "next" operator, $\ominus$ is the standard temporal "previous", U is the standard temporal "until", and S ("since") is a backwards-looking version of until. We assume the propositions include $\top$ (true) and $\bot$ (false), with their usual meanings, and define as abbreviations the derived operators "eventually ϕ" ($\Diamond\phi \equiv true\,\mathsf{U}\,\phi$), and "always ϕ" ($\Box\phi \equiv \neg\Diamond\neg\phi$), and similar backwards-looking versions $\diamondsuit\phi \equiv true\,\mathsf{S}\,\phi$ and $\boxminus\phi \equiv \neg\diamondsuit\neg\phi$. In some literature $\Diamond$ is denoted by $\mathbf{F}$, $\Box$ by $\mathbf{G}$, $\diamondsuit$ by $\mathbf{F}^-$ and $\boxminus$ by $\mathbf{G}^-$.

The remaining cases are standard in hybrid logic [6]: we have so-called *state variables*, with typical element x, which can be bound to nominals, and we have nominals n. A nominal is viewed as a logical proposition that is true in exactly one state, i.e. the state "designated" by the nominal. The operator $@_t\phi$, where t is either a state variable

$$
\begin{array}{rll}
\mathcal{M}, g, i \models p & \text{iff} & m_i \in V(p) \\
\mathcal{M}, g, i \models \neg\phi & \text{iff} & \mathcal{M}, g, i \not\models \phi \\
\mathcal{M}, g, i \models \phi_1 \wedge \phi_2 & \text{iff} & \mathcal{M}, g, i \models \phi_1 \text{ and } \mathcal{M}, g, i \models \phi_2 \\
\mathcal{M}, g, i \models \bigcirc\phi & \text{iff} & \mathcal{M}, g, i+1 \models \phi \\
\mathcal{M}, g, i \models \ominus\phi & \text{iff} & \mathcal{M}, g, i-1 \models \phi \\
\mathcal{M}, g, i \models \phi_1 \,\mathsf{U}\, \phi_2 & \text{iff} & \exists k \geq i : \mathcal{M}, g, k \models \phi_2 \text{ and } \forall j \text{ such that } i \leq j < k : \mathcal{M}, g, j \models \phi_1 \\
\mathcal{M}, g, i \models \phi_1 \,\mathsf{S}\, \phi_2 & \text{iff} & \exists k \leq i : \mathcal{M}, g, k \models \phi_2 \text{ and } \forall j \text{ such that } i \geq j > k : \mathcal{M}, g, j \models \phi_1 \\
\mathcal{M}, g, i \models x & \text{iff} & m_i = g(x) \\
\mathcal{M}, g, i \models n & \text{iff} & V(n) = \{m_i\} \\
\mathcal{M}, g, i \models @_t\phi & \text{iff} & \mathcal{M}, g, j \models \phi \text{ where } V(t) = \{m_j\} \text{ if } t \text{ is a nominal} \\
 & & \text{and } m_j = g(t) \text{ if } t \text{ is a state variable.} \\
\mathcal{M}, g, i \models \downarrow x\phi & \text{iff} & \mathcal{M}, g[x \mapsto m_i], i \models \phi \\
\mathcal{M}, g, i \models \mathsf{E}\phi & \text{iff} & \text{there exists } j \text{ s.t. } m_j \in \mathcal{M} \text{ and } \mathcal{M}, g, j \models \phi
\end{array}
$$

Fig. 2. Infinite-path semantics of the logic

or a nominal, shifts evaluation to the state t and can be read as "ϕ holds in state t". The operator $\downarrow x\phi$ binds the state variable x to the current state. Finally, the existential modality $\mathsf{E}\,\phi$ says that there exists a state in which ϕ holds, and its dual is the universal modality A. The use of nominals is important: we rely on each state having a unique label in order to define our Exp modality (see Section 5.3).

The formal semantics for this logic is given in Fig. 2 with respect to a hybrid Kripke structure $\mathcal{M}$, which consists of an infinite sequence of states $\langle m_1, m_2 \ldots\rangle$ and a valuation function V that maps propositions and nominals to the set of states in which they hold, i.e. $\mathcal{M} = \langle\langle m_1, m_2 \ldots\rangle, V\rangle$. We use the index i to refer to state m_i. The function g maps state variables x to states, and we write $g[x \mapsto m_i]$ to denote the function that maps x to m_i and otherwise behaves like g. Note that the rules of Fig. 2 only apply for $i \geq 1$. For $i < 1$ we have $\mathcal{M}, g, i \not\models \phi$.

When evaluating whether an expectation is fulfilled in a state m_i we want to not only determine whether the formula holds, but also whether an agent in state m_i is able to conclude that the formula holds. For example, if p is true in m_2, then even through $\bigcirc p$ holds in m_1, an agent in m_1 would not normally be able to conclude this, since it cannot see into the future.

We deal with this by using a simplified form of the operator $\mathsf{Trunc_S}$ from Eisner *et al.* [7]. A formula $\mathsf{Trunc_S}\,\phi$ is true at a given state in a model if and only if ϕ can be shown to hold without any knowledge of future states. We define this formally as:

$$\mathcal{M}, g, i \models \mathsf{Trunc_S}\,\phi \quad \text{iff} \quad \mathcal{M}^i, g, i \models^{+} \phi$$

where $\models^{+}$ represents the *strong semantics* of Eisner et al. (defined below), and $\mathcal{M}^i$ is defined as follows. Let $\mathcal{M} = \langle\langle m_1 \ldots m_i \ldots\rangle, V\rangle$. We define $V^i(p) = V(p) \backslash \{m_{i+1} \ldots\}$, that is, V^i gives the same results as V, but without states m_j for $j > i$. We then define $\mathcal{M}^i = \langle\langle m_1 \ldots m_i\rangle, V^i\rangle$. We write $i > |\mathcal{M}|$ to test for states that have been pruned, i.e. if $i > |\mathcal{M}|$ then there is no m_i in $\mathcal{M}$. We write $i \leq |\mathcal{M}|$ to test for states that

Condition	$\mathcal{M}, g, i \models^{+} \phi$?	$\mathcal{M}, g, i \models^{-} \phi$?
$i > \lvert\mathcal{M}\rvert$	false	true
$i \leq \lvert\mathcal{M}\rvert$ and $\phi = \neg\psi$	iff $\mathcal{M}, g, i \not\models^{-} \psi$	iff $\mathcal{M}, g, i \not\models^{+} \psi$
otherwise	As for $\models$, but substitute $\models^{+}$ or $\models^{-}$ (respectively) for $\models$ in recursive definitions	

Fig. 3. Strong and weak semantics on finite paths

do exist, i.e. if $i \leq |\mathcal{M}|$ then $m_i \in M$ (where $\mathcal{M} = \langle M, V \rangle$). We need to use the strong semantics ($\models^{+}$) because the standard semantics is defined over infinite sequences of states and does not provide any way to disregard information from future states. The strong semantics is skeptical: it concludes that $\mathcal{M}, g, i \models^{+} \phi$ only when there is enough evidence so far to definitely conclude that ϕ holds. To define negation, we also need its *weak* counterpart, $\models^{-}$. The weak semantics is generous: it concludes that $\mathcal{M}, g, i \models^{-} \phi$ whenever there is no evidence against ϕ so far.

Fig. 3 defines the strong and weak semantics. Note that the semantics of negation switch between the strong and weak semantics: we can conclude *strongly* (respectively weakly) that $\neg\phi$ holds if and only if we can conclude *weakly* (respectively strongly) that ϕ does not hold.

We can now use the $\mathsf{Trunc}_{\mathsf{S}}$ operator to define fulfilment and violation:

$$\mathsf{Fulf}\,\phi \equiv \mathsf{Exp}\,\phi \wedge \mathsf{Trunc}_{\mathsf{S}}\,\phi$$

$$\mathsf{Viol}\,\phi \equiv \mathsf{Exp}\,\phi \wedge \mathsf{Trunc}_{\mathsf{S}}\,\neg\phi$$

4 Formula Progression

As outlined in Section 2, we use the notion of *formula progression* to describe how an unfulfilled and non-violated expectation evolves from one state to the next. Formula progression was introduced in the TLPlan planner to allow "temporally extended goals" to be used to control the system's search for a plan. Rather than just describing the desired goal state for the plan to bring about, TLPlan used a linear temporal logic formula to constrain the path of states that could be followed while executing the plan. As planning proceeds, whenever a new action is appended to the end of the plan, the goal formula must be "progressed" to represent the residual constraint left once planning continues from the state resulting from executing that action.

Bacchus and Kabanza considered progression as a function mapping a formula and state to another formula, and defined this function inductively on the structure of formulae in their logic $\mathcal{LT}$—a first-order version of LTL. They proved the following theorem.

Theorem (Bacchus and Kabanza [5]). *Let $M = \langle w_0, w_1, \ldots \rangle$ be any $\mathcal{LT}$ model. Then, we have for any $\mathcal{LT}$ formula f in which all quantification is bounded, $\langle M, w_i \rangle \models f$ if and only if $\langle M, w_{i+1} \rangle \models \mathit{Progress}(f, w_i)$.*

In other words, the truth of a linear temporal logic formula at a given point on a history of states is equivalent to the truth of the progressed formula at the next state in

$$\mathcal{M}, g, i \models \mathsf{Progress}(p, \psi) \text{ where } \begin{cases} \psi = \top & \text{if } p \in V(m_i) \\ \psi = \bot & \text{otherwise} \end{cases}$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\phi_1 \wedge \phi_2, \psi_1 \wedge \psi_2) \text{ iff } \mathcal{M}, g, i \models \mathsf{Progress}(\phi_1, \psi_1) \text{ and } \mathcal{M}, g, i \models \mathsf{Progress}(\phi_2, \psi_2)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\neg\phi, \neg\psi) \text{ iff } \mathcal{M}, g, i \models \mathsf{Progress}(\phi, \psi)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\bigcirc\phi, \phi)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\phi_1 \,\mathsf{U}\, \phi_2, \psi_2 \vee (\psi_1 \wedge (\phi_1 \,\mathsf{U}\, \phi_2))) \text{ iff } \mathcal{M}, g, i \models \mathsf{Progress}(\phi_1, \psi_1) \text{ and } \mathcal{M}, g, i \models \mathsf{Progress}(\phi_2, \psi_2)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\ominus\phi, \ominus\ominus\phi)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\phi_1 \,\mathsf{S}\, \phi_2, \ominus(\phi_1 \,\mathsf{S}\, \phi_2))$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(x, \psi) \text{ where } \begin{cases} \psi = \top & \text{if } m_i = g(x) \\ \psi = \bot & \text{otherwise} \end{cases}$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(n, \psi) \text{ where } \begin{cases} \psi = \top & \text{if } V(n) = \{m_i\} \\ \psi = \bot & \text{otherwise} \end{cases}$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\downarrow x\phi, \psi) \text{ iff } \mathcal{M}, g, i \models \mathsf{Progress}(\phi[x/n], \psi) \text{ where } V(n) = \{m_i\}$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(@_t\phi, @_t\phi)$$

$$\mathcal{M}, g, i \models \mathsf{Progress}(\mathsf{E}\phi, \mathsf{E}\phi)$$

Fig. 4. Recursive evaluation of the progression operator

the history. This provides an incremental way of evaluating future-oriented temporal formulae.

In the theorem above, $Progress(f, w_i)$ is a meta-logical function. We wish to define progression as an operator within the logic, and so adapt the above theorem to provide a *definition* of the modal operator $\mathsf{Progress}(\phi, \psi)$:

$$\mathcal{M}, g, i \models \mathsf{Progress}(\phi, \psi) \text{ iff } \forall \mathcal{M}' \in \overline{\mathcal{M}}(i),\ \mathcal{M}', g, i \models \phi \iff \mathcal{M}', g, i{+}1 \models \psi$$

where $\overline{\mathcal{M}}(i)$ is the set of all possible infinite models that are extensions of $\mathcal{M}^i$ ($\mathcal{M}$ truncated at i) and which preserve all the nominals in $\mathcal{M}$ (including those at indices past i). Apart from the requirement to agree on nominals, the models of $\overline{\mathcal{M}}(i)$ need not agree with $\mathcal{M}$ on the truth of propositions for state indices $j > i$.

We can then obtain the theorems of Fig. 4, which define an inductive procedure for evaluating progression, in conjunction with the use of Boolean simplification to eliminate $\bot$ and $\top$ as subformulae. This procedure is similar to the function of Bacchus and Kabanza, but extended to account for the hybrid features of our logic. The theorem for the binder operator requires there to be a nominal naming the state m_i ($\phi[x/n]$ denotes subsitution of the nominal n for the free occurrences of x); however, for our

model checking application, this can be easily ensured by preprocessing the model to add nominals for states that lack them.

5 Applying Model Checking to Expectation Monitoring

Model checking is the problem of determining for a *particular* model of a logical language whether a given formula holds in that model. Thus it differs from logical inference mechanisms which make deductions based on rules that are valid in *all* possible models. This makes model checking more tractable in general than deduction.

Model checking is commonly used for checking that models of dynamic systems, encoded as finite state machines, satisfy properties expressed in a temporal logic. However, model checking is also able to check paths, and we have therefore investigated the applicability of model checking as a way of checking for expectations, fulfilments and violations over a model which is a linear history of observed states. This was done by extending an existing model checker, described in the next section.

5.1 The Hybrid Logics Model Checker

The Hybrid Logics Model Checker (HLMC) [2] implements the MCLITE and MCFULL labelling algorithms of Franceschet and de Rijke [8]. HLMC reads a model encoded in XML and a formula given in a textual notation, and uses the selected labelling algorithm to determine the label, true ($\top$) or false ($\bot$), for the input formula in each state of the model. It then reports to the user all the states in which the formula is true (i.e. it is a *global* model checker).

The two labelling algorithms are defined over a propositional temporal logic with the operators **F** ("some time in the future"), **P** ("some time in the past"), the binary temporal operators **U** (until) and **S** (since), the universal modality **A**, and the following features of hybrid logic: nominals, state variables, the operator $@_t$, and the binding operators $\downarrow x$ and $\exists x$ ("binding x to some state makes the following expression true"). The duals of the modal operators are defined in the usual way. The underlying accessibility relation is assumed to represent "later than" (the transitive closure of the "next state" relation underlying many temporal logics), so the definitions of **F** and **P** in terms of the accessibility relation are equivalent to those of our $\bigcirc$ and $\ominus$, respectively, over a "next state" accessibility relation. Time is not constrained to be linear.

The global model checking problem for any subset of this language that freely combines temporal operators with binders is known to be PSPACE-complete [8]. MCLITE is a bottom-up labelling algorithm for the sublanguage that excludes the two binding operators, and it runs in time $O(k.n.m)$ where k is the length of the formula to be checked, n is the number of states in the model, and m is the size of the model's accessibility relation. MCFULL handles the full language, uses polynomial space, and runs in time exponential on the nesting degree of the binders in the formula.

MCLITE works by labelling each subformula of the formula to be checked, for all states in the model, in a bottom-up manner. Fig. 5 illustrates this process for an example formula. For each subformula, its Boolean value in each state (its *label*) is calculated, and a labelling function for the parent formula's outermost operator is then used to generate the label for that formula as a function of the subformulae's labels. Fig. 6 shows

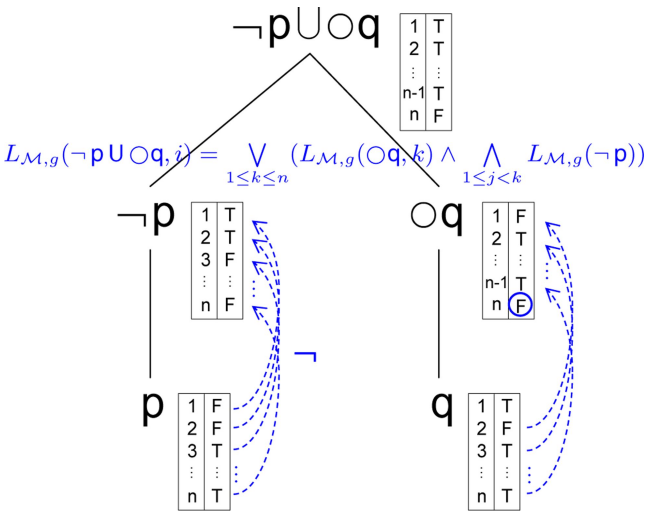


Fig. 5. Example label computation in HLMC

the semantics of some of the operators supported by HLMC together with the corresponding definition of the label, denoted $L_{\mathcal{M},g}(\phi, i)$. The presentation is adapted from that of Franceschet and de Rijke [8] to correspond to the HLMC operators, and to provide a declarative rather than procedural account[1]. We use $[V, g](a)$ as an abbreviation for either the value of $V(a)$ if a is a nominal or $\{g(a)\}$ if a is a state variable. It can be seen that in these cases the labelling function is a straightforward translation from the semantics—a property we have sought to preserve where possible for our extended notion of labels presented in Section 5.2.

The simple bottom-up procedure does not work when binders are included in the language as there will be subformulae containing free state variables, and the values of these depend on the enclosing binding context. Instead, the recursive top-down MCFULL procedure is used. A formula is labelled by first labelling its immediate subformulae recursively, and then applying the appropriate labelling algorithm for the formula's operator. For operators in the MCLITE sublanguage, the MCLITE labelling algorithm is used. When the recursion encounters a formula of the form $\downarrow x \phi_x$, the recursive labelling is performed for *each* binding of x to a state in the model (consider the formula $\mathbf{G} \downarrow x \, @_x p$: labelling this for any given state s requires the truth of $@_x p$ to be known for all bindings of x to future states s).

5.2 Handling $\mathsf{Trunc_S}$

We have adapted HLMC for checking the fulfilment and violation of expectations. We assume (and verify) that the input model represents a linear path and thus contains a single "next state" accessibility relationship.

[1] For consistency with the rest of the paper we use the notation defined in Section 3 and assume that models are sequences of states. HLMC does not, in fact, restrict models to be sequences, but for our application to monitoring observed traces this is a valid restriction.

Nominals and state variables

$$\mathcal{M}, g, i \models a \text{ iff } m_i \in [V, g](a)$$

$$L_{\mathcal{M},g}(a, i) = \begin{cases} \top & \text{if } m_i \in [V, g](a) \\ \bot & \text{otherwise} \end{cases}$$

Operator $@_t$

$$\mathcal{M}, g, i \models @_t\phi \text{ iff } \mathcal{M}, g, j \models \phi \text{ where } [V, g](t) = \{m_j\}$$

$$L_{\mathcal{M},g}(@_t\phi, i) = L_{\mathcal{M},g}(\phi, j) \text{ where } [V, g](t) = \{m_j\}$$

Operator $\bigcirc$

$$\mathcal{M}, g, i \models \bigcirc\phi \text{ iff } i < |\mathcal{M}| \wedge \mathcal{M}, g, i+1 \models \phi$$

$$L_{\mathcal{M},g}(\bigcirc\phi, i) = (i < |\mathcal{M}| \wedge L_{\mathcal{M},g}(\phi, i+1))$$

Fig. 6. The MCLITE labelling function (partial definition)

To allow the checking of fulfilment and violation a labelling algorithm for $\mathsf{Trunc_S}$ was developed. This was complicated by the presence of past-time operators. Consider the label for $\mathsf{Trunc_S} \ominus\neg\phi$. Based on the definitions of Section 3, we have:

$$\begin{aligned} \mathcal{M}, g, i \models \mathsf{Trunc_S} \ominus\neg\phi &\iff \mathcal{M}^i, g, i \models^+ \ominus\neg\phi \\ &\iff \mathcal{M}^i, g, i{-}1 \models^+ \neg\phi \\ &\iff \mathcal{M}^i, g, i{-}1 \not\models \phi \end{aligned}$$

or equivalently:

$$\begin{aligned} L_{\mathcal{M},g}(\mathsf{Trunc_S} \ominus\neg\phi, i) &= L^+_{\mathcal{M}^i,g}(\ominus\neg\phi, i) \\ &= L^+_{\mathcal{M}^i,g}(\neg\phi, i{-}1) \\ &= \neg L^-_{\mathcal{M}^i,g}(\phi, i{-}1) \end{aligned}$$

where $L^+_{M,g}$ and $L^-_{M,g}$ denote labelling under the strong and weak semantics, respectively. Thus to label $\mathsf{Trunc_S} \ominus\neg\phi$ at model index i it is necessary to know the weak semantics label for ϕ at index $i{-}1$ when the model is truncated at i. More generally, when labelling a formula ϕ at a model index i it is necessary to store both weak and strong labels with respect to all possible future truncation points: $L^-_{M^j,g}(\phi, i)$ and $L^+_{M^j,g}(\phi, i)$ for $j \geq i$. We therefore define a generalised label for a formula ϕ at model index i as a sequence of pairs of weak and strong labels for each possible truncation point from i to the final state in the model:

$$L_{\mathcal{M},g}(\phi, i) = \left\langle \left(L^-_{\mathcal{M}^j,g}(\phi, i),\ L^+_{\mathcal{M}^j,g}(\phi, i) \right) \mid i \leq j \leq |\mathcal{M}| \right\rangle$$

where $L^-_{\mathcal{M}^j,g}(\phi, i)$ is the value for ϕ at index i in the model under the weak semantics assuming a truncation at index j, and $L^+_{\mathcal{M}^j,g}(\phi, i)$ is the corresponding value under the strong semantics. Note that $L_{\mathcal{M},g}(\phi, i)$ is an empty sequence for $i > |\mathcal{M}|$.

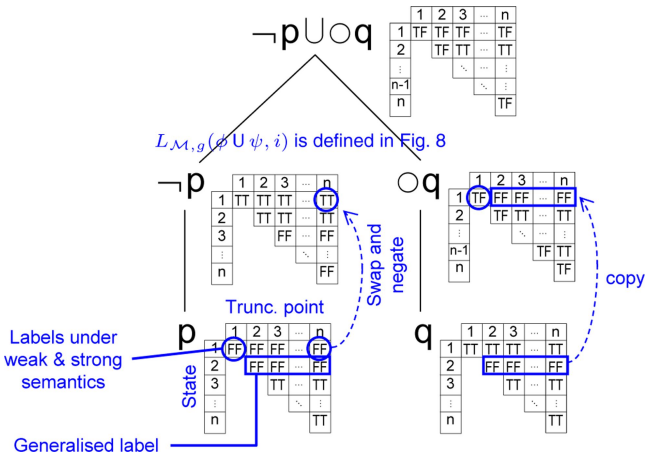


Fig. 7. Example use of generalised labels

Figure 7 illustrates the representation and calculation of generalised labels. The figure shows generalised labels as rows in diagonal matrices that are indexed by the state at which evaluation is to be done and a possible (future) truncation point for the model. However, we represent and compute with each generalised label $L_{\mathcal{M},g}(\phi, i)$ in an abbreviated form that can omit weak/strong value pairs if there has been no change in these values since the previous truncation point in the sequence. Each entry in the sequence is a truncation point associated with a pair of weak and strong values:

$$L_{\mathcal{M},g}(\phi, i) = \langle j_1 : (w_{j_1}, s_{j_1}), \ldots, j_n : (w_{j_n}, s_{j_n}) \rangle$$

where $i = j_1 < \cdots < j_n < j_{n+1} = |\mathcal{M}| + 1$ and $\forall_{1 \le k \le n} \forall_{j_k \le l < j_{k+1}} L^{-}_{\mathcal{M}^l,g}(\phi, i) = w_{j_k} \wedge L^{+}_{\mathcal{M}^l,g}(\phi, i) = s_{j_k}$.

Conjunction and disjunction apply to generalised labels in a straightforward way, acting element-wise, while negation operates on the arguments and then exchanges the resulting weak and strong values at each truncation point, e.g. $\neg \langle 1 : (\top, \bot), 2 : (\top, \top) \rangle = \langle 1 : (\top, \bot), 2 : (\bot, \bot) \rangle$. When $\wedge$ and $\vee$ are applied to labels $l = \langle i : (w_i, s_i), \ldots \rangle$ and $l' = \langle j : (w'_i, s'_i), \ldots \rangle$ where $i < j$, l' is treated as if it had $i : (\bot, \bot)$ prepended for $\vee$ and $i : (\top, \top)$ prepended for $\wedge$ (and l is treated similarly if $i > j$), i.e. the sequence starting at a later truncation point is padded with default weak and strong labels for truncation points i to $j-1$. We write indexed conjunctions and disjunctions, e.g. ${}^{i}\wedge_{1 \le k \le |\mathcal{M}|}$, with a prefix superscript index i, indicating that the value if there are no conjuncts or disjuncts is $\langle i : (\top, \top) \rangle$ or $\langle i : (\bot, \bot) \rangle$ respectively.

For the HLMC operators that are not future oriented, the declarative specifications of the MCLITE labelling functions (as shown in part in Fig. 6) can then be applied to these generalised labels. Labels for the temporal operators are computed using the definitions in Fig. 8. We also support the derived operators $\Diamond$, $\Box$, $\diamondminus$ and $\boxminus$ defined in Section 3. In the definition of the labelling function for U, the disjunct on the right captures the intuition that under the weak semantics $\phi \mathsf{U} \psi$ is satisfied if ϕ holds weakly up to the

Out of bound indices

$$L_{\mathcal{M},g}(\phi,i) = \begin{cases} \langle 1 : (\bot,\bot)\rangle & \text{for } i < 1 \\ \langle\rangle & \text{for } i > |\mathcal{M}| \end{cases}$$

Operators $\bigcirc$ and $\ominus$ $(1 \le i \le |\mathcal{M}|)$

$$L_{\mathcal{M},g}(\bigcirc\phi,i) = i : (\top,\bot) \bullet L_{\mathcal{M},g}(\phi, i{+}1)$$
where $\bullet$ is the prepend operation

$$L_{\mathcal{M},g}(\ominus\phi,i) = L_{\mathcal{M},g}(\phi, i{-}1) \restriction i$$
where $\sigma \restriction i$ is $\langle j : (w_j, s_j) \in \sigma \mid j \ge i\rangle$

Operators U and S $(1 \le i \le |\mathcal{M}|)$

$$L_{\mathcal{M},g}(\phi \,\mathsf{U}\, \psi, i) = \bigvee^{i}_{i \le k \le |\mathcal{M}|} \Big(L_{\mathcal{M},g}(\psi,k) \wedge \bigwedge^{k}_{i \le j < k} L_{\mathcal{M},g}(\phi,j) \Big) \vee \Big(\pi^w_i \wedge \bigwedge^{i}_{i \le j \le |\mathcal{M}|} L_{\mathcal{M},g}(\phi,j) \Big)$$
where $\pi^w_i = \langle i : (\top,\bot)\rangle$

$$L_{\mathcal{M},g}(\phi \mathsf{S} \psi, i) = \Big(\bigvee^{1}_{1 \le k \le i} \big(L_{\mathcal{M},g}(\psi,k) \wedge \bigwedge^{k}_{k < j \le i} L_{\mathcal{M},g}(\phi,j) \big) \Big) \restriction i$$

Fig. 8. Labelling temporal formulae in extended HLMC

end of the (finite) model, even if ψ never holds. The constant label π^w_i is used as a mask to ensure that this disjunct only applies for the weak semantics.

5.3 Defining Expectation, Fulfilment and Violation

We now show how the semantics of expectation, fulfilment and violation can be encoded within the extended HLMC. We elaborate on the intuitive account of these notions given in Section 2. We wish to use the model checker to check for the existence of rule-based conditional expectations, and their fulfilments and violations without requiring the rules of expectation to be hard-coded in the model checker, or integrated into the labelling procedure dynamically. Therefore, we define a *hypothetical* expectation modality $\mathsf{Exp}(\lambda, \rho, n, \phi)$. This means (informally) that if there *were* a rule $\lambda \to \mathsf{Exp}(\rho)$ then λ would have been strongly true at a previous state named by nominal n, the rule would have fired, and the expectation ρ would have progressed (possibly over multiple intermediate states) to ϕ in the current state. This means that we do not have to hard-code rules into the model checker, or provide a mechanism to read and internalise them. Instead, a rule of interest to the user can be supplied as arguments to an input formula using the ExistsExp modality. It is defined as follows.

$$\begin{aligned} \mathcal{M},g,i \models^{\pm} \mathsf{Exp}(\lambda,\rho,n,\psi) \text{ iff } & \mathcal{M},g,i \models^{\pm} \mathsf{Trunc}_{\mathsf{S}}\,\lambda,\ V(n)=\{m_i\} \text{ and } \psi=\rho \\ \text{or } & \exists\phi \text{ s.t. } \mathcal{M},g,i{-}1 \models^{\pm} \mathsf{Exp}(\lambda,\rho,n,\phi), \\ & \mathcal{M},g,i{-}1 \not\models^{\pm} \mathsf{Trunc}_{\mathsf{S}}\,\phi, \\ & \mathcal{M},g,i{-}1 \not\models^{\pm} \mathsf{Trunc}_{\mathsf{S}}\,\neg\phi \text{ and} \\ & \mathcal{M},g,i{-}1 \models^{\pm} \mathsf{Progress}(\phi,\psi) \end{aligned}$$

where we write $\models^{\pm}$ to indicate that the choice between the weak or strong semantics is immaterial as future states play no role in this definition.

The first conjunct in the definition expresses the case in which the hypothetical rule matches the current state. Note that we use $\mathsf{Trunc_S}$ when evaluating the rule's condition λ to restrict it to present and past information only. The second conjunct expresses the case of progressing a non-fulfilled and non-violated exectation from the previous state. Note that in order to use nominals to name the state at which rules apply, we require that the input model has been annotated with nominals for each state.

We also define hypothetical versions of Fulf and Viol as follows:

$$\mathcal{M}, g, i \models^{\pm} \mathsf{Fulf}(\lambda, \rho, n, \phi) \text{ iff } \mathcal{M}, g, i \models^{\pm} \mathsf{Exp}(\lambda, \rho, n, \phi) \text{ and } \mathcal{M}, g, i \models^{\pm} \mathsf{Trunc_S}\ \phi$$

$$\mathcal{M}, g, i \models^{\pm} \mathsf{Viol}(\lambda, \rho, n, \phi) \text{ iff } \mathcal{M}, g, i \models^{\pm} \mathsf{Exp}(\lambda, \rho, n, \phi) \text{ and } \mathcal{M}, g, i \models^{\pm} \mathsf{Trunc_S}\ \neg\phi$$

These modalities are not used directly by the model checker. Instead we define the following existential version of Exp:

$$\mathcal{M}, g, i \models^{\pm} \mathsf{ExistsExp}(\lambda, \rho) \text{ iff } \exists n, \phi \text{ s.t. } \mathcal{M}, g, i \models^{\pm} \mathsf{Exp}(\lambda, \rho, n, \phi)$$

with similar definitions for $\mathsf{ExistsFulf}(\lambda, \rho)$ and $\mathsf{ExistsViol}(\lambda, \rho)$. These correspond to the actual queries that we wish to make to the model checker: "are there any expectations (or fulfilments or violations) for the given rule, at any state in the model?"

To compute labels for these existential modalities, we first compute the following *witness function* $W_{\mathcal{M},g,i}$ iteratively for i increasing from 1 to $|\mathcal{M}|$ (where labels for the subformulae λ and ρ have already been computed due to HLMC's top-down recursive algorithm):

$$W_{\mathcal{M},g,i}(\mathsf{ExistsExp}(\lambda, \rho)) = \left\{ \begin{array}{ll} \{(n, \rho)\} & \text{where } V(n) = \{m_i\} \\ & \text{if } \mathcal{M}, g, i \models^{\pm} \mathsf{Trunc_S}\ \lambda \\ \varnothing & \text{otherwise} \end{array} \right\} \cup$$
$$\{(n, \psi) \mid \exists \phi. (n, \phi) \in W_{\mathcal{M},g,i-1}(\mathsf{ExistsExp}(\lambda, \rho)),$$
$$\mathcal{M}, g, i-1 \not\models^{\pm} \mathsf{Trunc_S}\ \phi,$$
$$\mathcal{M}, g, i-1 \not\models^{\pm} \mathsf{Trunc_S}\ \neg\phi \text{ and}$$
$$\mathcal{M}, g, i-1 \models^{\pm} \mathsf{Progress}(\phi, \psi)\}$$

This collects all pairs (n, ϕ) making $\mathsf{Exp}(\lambda, \rho, n, \phi)$ true at i for a given λ and ρ. The corresponding label for this formula at i is then $\langle i : (\bot, \bot) \rangle$ if the witness set is empty, and otherwise $\langle i : (\top, \top) \rangle$. Note that the generalised labels discussed in Section 5.2 are necessary for evaluating the $\mathsf{Trunc_S}$ formulae.

Witness functions are also defined for $\mathsf{ExistsFulf}(\lambda, \rho)$ and $\mathsf{ExistsViol}(\lambda, \rho)$ by taking the subset of pairs (n, ϕ) in $\mathsf{ExistsExp}(\lambda, \rho)$ for which $\mathsf{Trunc_S}\ \phi$ strongly holds and $\mathsf{Trunc_S}\ \neg\phi$ strongly holds, respectively.

Finally, we can use the extended HLMC to check for expectations, violations and fulfilments over a given model by performing the global model checking procedure with an empty intial binding g for an input formula such as $\mathsf{ExistsExp}(\lambda, \rho)$, $\mathsf{ExistsFulf}(\lambda, \rho)$ or $\mathsf{ExistsViol}(\lambda, \rho)$, where condition λ and expectation ρ correspond to some rule of interest. The model checker will report all witnesses for the input formula for all states.

This can be easily generalised to apply to disjunctions of input formulae referring to multiple rules. Note that although the witnesses for the ExistsExp modality could be used to generate labels for Exp for a given rule, we do not currently support the use of Exp to appear within rules, and so cannot handle interdependent expectations.

6 An Example

Consider the scenario shown in Fig. 1 of Section 2 (Scenario 1) and the modified scenario (Scenario 2) in which o is also true in s_2, i.e. a second order is placed before payment for the first has been received. In Scenario 2, the expectation created in s_1 and progressed to s_2 is violated in that state, whereas there is no violation in Scenario 1. The following witness lists are output from the model checker for these scenarios, given the input formulae $\mathsf{ExistsExp}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$, $\mathsf{ExistsFulf}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$ and $\mathsf{ExistsViol}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$. The list of witnesses (pairs) beside each state record the current existing, fulfilled or violated expectations (depending on the input formula), alongside a nominal naming the state in which the expectation was created. For example, in the right hand column for Scenario 2 we can see that at state s_2 the expection of $\neg o \,\mathsf{U}\, p$, created (in a more complex form) in state s_1 and progressed to s_2, is now violated.

Scenario 1

$\mathsf{ExistsExp}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$	$\mathsf{ExistsFulf}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$	$\mathsf{ExistsViol}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$
s_1: $(s_1, \bigcirc(\neg o \,\mathsf{U}\, p))$	s_1:	s_1:
s_2: $(s_1, \neg o \,\mathsf{U}\, p)$	s_2:	s_2:
s_3: $(s_1, \neg o \,\mathsf{U}\, p)$	s_3: $(s_1, \neg o \,\mathsf{U}\, p)$	s_3:
s_4: $(s_4, \bigcirc(\neg o \,\mathsf{U}\, p))$	s_4:	s_4:

Scenario 2

$\mathsf{ExistsExp}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$	$\mathsf{ExistsFulf}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$	$\mathsf{ExistsViol}(o, \bigcirc(\neg o \,\mathsf{U}\, p))$
s_1: $(s_1, \bigcirc(\neg o \,\mathsf{U}\, p))$	s_1:	s_1:
s_2: $(s_2, \bigcirc(\neg o \,\mathsf{U}\, p))$, $(s_1, \neg o \,\mathsf{U}\, p)$	s_2:	s_2: $(s_1, \neg o \,\mathsf{U}\, p)$
s_3: $(s_2, \neg o \,\mathsf{U}\, p)$	s_3: $(s_2, \neg o \,\mathsf{U}\, p)$	s_3:
s_4: $(s_4, \bigcirc(\neg o \,\mathsf{U}\, p))$	s_4:	s_4:

7 Related Work

There have been a variety of approaches to modelling expectations and commitments formally, some of which are outlined below.

Alberti et al. [9] represent conditional expectations as rules with an E modality in their conclusion. Abductive inference is used to generate expectations, which are monitored at run time. The temporal aspects of expectations are restricted to constraint logic programming constraints relating variables representing the time of events.

Verdicchio and Colombetti [10] use a first order variant of CTL* with past-time operators to provide axioms defining the lifecycle of commitments in terms of primitives

representing the existence, the fulfilment, and the violation of a commitment in a state. In their approach, commitments are always expressed from the viewpoint of the state in which they were created, and the formula $Comm(e, a, b, u)$, recording that event e created a commitment from a to b that u holds, remains true in exactly that form from one state to the next. Fulfilment is then defined by a temporal formula that searches back in time for the event that created the commitment, and then evaluates the content u at that prior state, for all paths passing through the current state.

Bentahar et al. [11] present a logical model for commitments based on a branching time temporal logic in the context of semantics for argumentation. The semantics use accessibility relations for different types of commitments. These encode deadlines that are associated with commitments on their creation.

Model checking has been applied to statically verifying properties of closed systems of agents (thus their programs are available to form the input model) and also for checking desired properties of institution specifications. A recent example of the latter is the work of Viganò and Colombetti [12]. Of more relevance to this paper is the application of model checking to run-time compliance checking based on observed traces.

Endriss [13] discussed the use of *generalised model checking* for deciding whether a trace of an agent dialogue conforms to a protocol expressed in propositional linear temporal logic.

Spoletini and Verdicchio [14] addressed the online monitoring of commitments expressed in a propositional temporal logic with both past and future operators. They proposed a distributed processing architecture that included formula analyser modules based on alternating automata. The discussion is procedural rather than declarative, and further analysis is needed to compare the technique to our approach.

8 Conclusions and Future Work

This paper has presented a logical account of the notions of conditional expectation, fulfilment and violation in terms of a linear temporal logic. For offline monitoring of expectations, the problem of determining fulfilment and violation of expectations without recourse to future information was identified as a key problem, and a solution was presented in terms of path truncation and the strong semantics of Eisner et al. [7]. It was then shown how the MCLITE and MCFULL model checking algorithms can be modified to support the truncation operator by using generalised labels that record for a model state the truth values under both the weak and strong semantics for all possible future states. An existing model checker (HLMC) has been modified using these techniques to allow the existence of expectations, and fulfilments and violations of these expectations to be detected.

A hybrid propositional temporal logic was used in this work as that is what was implemented by HLMC. Including nominals allowed our Exp modality to record the states in which expectations were created. However, with our focus on *linear* models, the other hybrid constructs have limited value for defining conditional expectations. We plan to extend our approach to apply to a real-time temporal logic interpreted over timed paths. In this case, binders and state variables become useful for expressing timing relations between states.

We also plan to investigate extending the technique to apply to some suitably constrained fragment of first order temporal logic (e.g. the guarded fragment). Other future work includes modifying the internal data structures and labelling algorithms to support incremental online monitoring of expectations, a detailed analysis of the complexity of the modified algorithms and empirical evaluations.

References

1. Cortés, U.: Electronic institutions and agents. AgentLink News 15, 14–15 (2004)
2. Dragone, L.: Hybrid logics model checker (2005), `http://luigidragone.com/hlmc/`
3. Cranefield, S.: A rule language for modelling and monitoring social expectations in multi-agent systems. In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM 2005 and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 246–258. Springer, Heidelberg (2006)
4. Markey, N., Schnoebelen, P.: Model checking a path. In: Amadio, R., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 251–265. Springer, Heidelberg (2003)
5. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artificial Intelligence 116(1-2), 123–191 (2000)
6. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
7. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Campenhout, D.V.: Reasoning with temporal logic on truncated paths. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 27–39. Springer, Heidelberg (2003)
8. Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). Journal of Applied Logic 4(3), 279–304 (2006)
9. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based software tool. In: Trappl, R. (ed.) Cybernetics and Systems 2004. Austrian Society for Cybernetics Studies, vol. II, pp. 570–575 (2004)
10. Verdicchio, M., Colombetti, M.: A logical model of social commitment for agent communication. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003), pp. 528–535. ACM Press, New York (2003)
11. Bentahar, J., Moulin, B., Meyer, J.J.C., Chaib-draa, B.: A logical model for commitment and argument network for agent communication. In: AAMAS 2004, pp. 792–799. IEEE Computer Society, Los Alamitos (2004)
12. Viganò, F., Colombetti, M.: Symbolic model checking of institutions. In: Proceedings of the 9th international conference on electronic commerce, pp. 35–44. ACM Press, New York (2007)
13. Endriss, U.: Temporal logics for representing agent communication protocols. In: Dignum, F.P.M., van Eijk, R.M., Flores, R. (eds.) AC 2005. LNCS (LNAI), vol. 3859, pp. 15–29. Springer, Heidelberg (2006)
14. Spoletini, P., Verdicchio, M.: An automata-based monitoring technique for commitment-based multi-agent systems. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 172–187. Springer, Heidelberg (2008)

The Use of Norms Violations to Model Agents Behavioral Variety

Benoit Lacroix[1,2], Philippe Mathieu[2], and Andras Kemeny[1]

[1] Renault, Technical Center for Simulation
Technocentre - 1 avenue du Golf
78288 Guyancourt Cedex, France
{benoit.lacroix,andras.kemeny}@renault.com
[2] LIFL - CNRS UMR 8022
University of Lille - Cite Scientifique Bat. M3
59655 Villeneuve d'Ascq Cedex, France
philippe.mathieu@lifl.fr

Abstract. In multi-agent applications, normative systems are usually used to regulate the behavior of the agents. They provide an efficient means to ensure limited deviations from an expected ideal behavior. Many works have been done in this classical research direction, less frequent are the works on norms in simulation. In this paper we focus on the simulation of spatially situated agents, typically moving around simulated physical environments. Our goal is to provide a mechanism allowing an efficient generation of consistent agents characteristics. We propose to model behavioral differentiation as violations of the norms, and show its application to traffic simulation with the driving simulation software used at Renault, SCANeR™.

1 Introduction

Many multi-agent applications benefit greatly from the notion of normative systems. Such applications can exploit many characteristics of norms: they offer regulation possibilities, and can help to introduce coordination and cooperation improvements. The field of application has thus grown during the last years from law and virtual societies to disaster management or transport, and is still widening. However, works mainly concern normative system architectures [1], norm representations [2], norm adherence, or norm emergence among societies [3]. Less common are works on norms in simulation.

Norms are usually used to specify the ideal behavior of the agents within the system. Indeed, the autonomy left to the agents tends to move them away from their ideal behavior. Normative systems provide an interesting regulation means: when the ideal behavior is considered as a norm, the objective is to make the agents comply with it. In Electronic Institutions [4,5,6] for instance, the institution uses norms to manage the social interactions of the agents. They interact within the environment, and the institution provides authority and control instances designed to regulate their behavior.

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 220–234, 2009.

Some works have used norm in the context of simulation of spatially situated agents by focusing on the regulation capabilities, and not on the organizational structure. Bou et al. [7] study how traffic control strategies are improved by extending Electronic Institutions with autonomic capabilities. Depending on traffic events, the institution optimizes its response, like the fines amount. In [8], the authors show how the introduction of non-normative behaviors improves the realism of microscopic traffic simulation. By allowing agents to break some of the rules of the road, norms are implicitly taken into account in the decision model.

To improve the realism of the model, violations are sometimes allowed, or even encouraged [9]. In such cases, the institution provides adapted sanctions to regulate agents behavior. We propose in this paper to describe the behaviors using norms, and to use violations to efficiently create realistic and diversified behaviors. The normative system is thus not considered as a regulation means of the agents internal state – as part of their decision model –, but as an environment's regulation means of the agents population.

This paper is organized as follows. First, we present the context of our study: the simulation of spatially situated agents. Then we describe the institutional environment, and present our approach: modeling behavioral differentiation in simulations as norms violations. Finally, the application of the model to the driving simulation software used at Renault, SCANeR™, is shown, and experimental results demonstrating the interest of the approach are presented.

2 Simulation of Spatially Situated Agents

2.1 The Need of Behavioral Variety

In this paper, we consider the application of normative systems in a specific context: the simulation of spatially situated agents. This kind of simulation includes all simulations where individual characteristics result in different behaviors, like for instance pedestrian simulations [10] or traffic simulations [11]. In such simulations, agents move around the environment: they need to be able to compute their positions and displacements. Besides, we consider here only microscopic simulation. Agents behavior may be observed continuously, and we have to ensure that each of their actions is realistic.

In this context, the variety of behaviors is important to be able to observe realistic situations during the simulation. Indeed, group phenomena can emerge from the microscopic interactions of the agents. These phenomena, observed in the real world, are for instance the formation of lines in multidirectional pedestrians flows, or the regrouping effects caused by the sociability of individuals (people tend to approach a group rather than staying alone). Even if all the agents own the same set of characteristics and use the same models (decision, displacement models), these phenomena might be observed in the simulation. However, the possibility to obtain individual behaviors, like people staying alone or with small groups, is not intrinsically guaranteed without complementary mechanisms.

Creating a behavioral variety is crucial for the simulation's realism. To achieve this goal we have to provide the agents with different individual characteristics:

for pedestrians it could be the size of the agents or the displacement model they use; for drivers the desired speed or the safety time.

2.2 The Need of Behavioral Consistency

Another point is that we have to be able to control the consistency of agents behaviors. Indeed, if the simulation produces inconsistent ones when it is designed to reproduce real world situations, the validity of the experimentation has to be reconsidered.

In most simulations, sets of parameters characterize agents behaviors. Any set can be generated and used, but only some of them result in meaningful behaviors: only those should be kept (Fig. 1). In Figure 2, a more specific example is presented. We suppose that drivers are characterized by two parameters, acceleration a and safety time t, which can be picked out from continuous predefined intervals. When generating randomly combinations of these parameters, drivers using a high acceleration and a low safety time are created, as well as drivers using a low acceleration and a high safety time. They can naturally be classified as aggressive and cautious, matching usual classifications of real drivers. However, other associations are also produced: drivers using high acceleration and safety time, or low ones. The behaviors resulting from these parameters are not realistic, and a mechanism has to be provided to exclude them.

To be able to introduce accurately proportion of agents showing specific and consistent behaviors, we need to be able to use only specific sets of parameters and to quantify their validity.

2.3 Towards a Normative Model

The description capabilities of norms offer various assets to achieve the different goals presented above. Indeed, they provide different means to create the variety

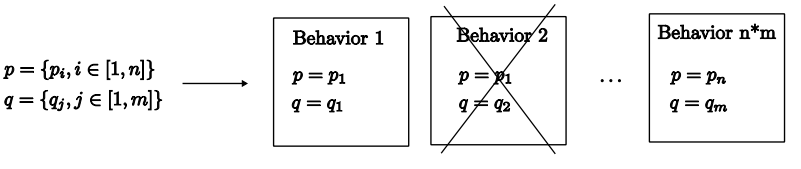


Fig. 1. Only some sets of parameters should be generated to produce consistent behaviors. A mechanism excluding inconsistent ones (like behavior 2) has to be provided.

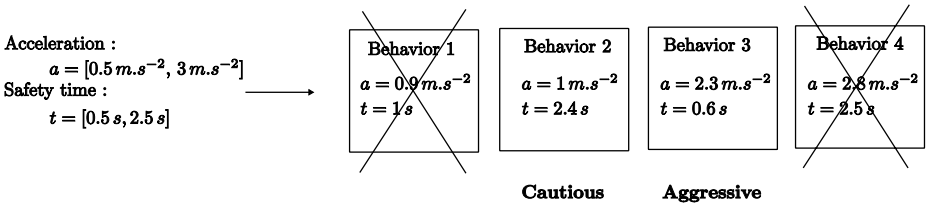


Fig. 2. A similar example using real parameters. The only sets of parameters we want to keep have to match meaningful behaviors.

we are looking for. The first possibility is to exploit the definitions of the norms themselves. They can be used as generic structures allowing describing any kind of behavior: a wide variety of norms may coexist, using various parameters. The second possibility is to allow violations of the defined norms, which can produce interesting new and unexpected behaviors. As for the consistency, the generation of the behaviors within the norms limits guarantees it. If violations are allowed, the deviations have to be quantified to remain in predefined limits.

Finally, when dealing with simulation of spatially situated agents, the goal is often to reproduce existing behaviors. The intuitive description of the world provided by norms allows users and scenario designers to easily comprehend the generation mechanism, which they can then configure and modify by themselves.

3 Institutional Environment

In our case, norms are used to build and control the context of the simulation, and not as the decision model of the agents. We do not use here explicit authoring structures: norms are only used to create agents characteristics.

3.1 Semantic

We made the choice to use the same terminology as in classical normative approaches, but voluntarily did not used the terms in their common acceptance. The definitions are adapted to the context, as this redefinition allows describing efficiently the model.

Institution. According to the choice we presented, the institution does not handle authority and controller agents. Its role is to manage the norms in the environment. However, the institution may be related to a particular context, so we keep track of sets of institutional and environmental properties. The institutional properties refer to criteria regarding law and obligations, environmental ones are related to contextual elements. The institution is mainly used as a set of parameters and definition domains. Parameter is used here with a wide meaning: it can be an action rule associated to its pre-conditions.

Definition 1. *We define an Institution as a tuple* $\langle P, D_P, P_i, P_e \rangle$ *where:*

- P *is a finite set of parameters.*
- $D_P = \{d_p, \forall p \in P\}$ *is a set of definition domains.*
- P_i *is a set of institutional properties.*
- P_e *is a set of environmental properties.*

Norm. Norms are defined as a subset of the institution parameters, associated to subsets of the definition domains. For instance, a norm can be described by a parameter and the distribution function describing the values it can take. Norms handle specific sets of institutional and environmental properties, which can specialize institution ones. Conflicting norms are allowed; their preference ordering

and their interpretation is left to the agents decision model. At this step, enforcement strategies, like punishment, are not included. Several norms can be defined for the same environment, and norms can have non-empty intersections.

Definition 2. *We define a Norm as a tuple* $\langle I, P_n, D_{n_{P_n}}, P_{n_i}, P_{n_e} \rangle$ *where:*

- I *is the institution the norm refers to.*
- $P_n \subset P$ *is the subset of parameters associated to the norm.*
- $D_{n_{P_n}} \subset D_P$ *is the subset of definition domains:* $\forall p_n \in P_n, \exists p \in P, p_n = p, d_{n_{p_n}} \subset d_p$
- P_{n_i} *is a set of institutional properties.*
- P_{n_e} *is a set of environmental properties.*

Behavior. A behavior describes the instantiation of a norm. Each element of the behavior is described by a parameter taken from the corresponding norm, and a value associated to this parameter. This value can be taken in or outside the definition domain associated to this parameter in the norm. Note that the definition domain can be a set of functions: the parameter's associated value will then be itself a function.

Definition 3. *A Behavior is defined as a tuple* $\langle N, P_b, V_{P_b} \rangle$ *where :*

- N *is a reference to the instantiated norm.*
- P_b *is a subset of the set of parameters defined in the instantiated norm.*
- V_{P_b} *is the set of values associated to the parameters.*

A detailed example using these definitions is presented in Section 4.4.

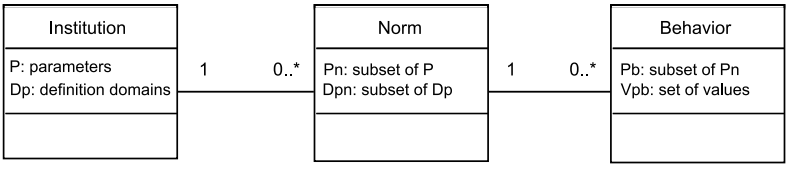


Fig. 3. The different elements of the institutional environment and their relationships

3.2 Behavioral Variety as Violation of the Norm

The violation of the norms offers possibilities to increase the behavioral variety. For each agent, a behavior is instantiated. The set of parameters and associated values is determined during the instantiation: they can either be in the definition domain defined by the norm, or outside. If the value is in the definition domain, the parameter respects the norm. If not, it is a violation. The norm being known, we are able to establish which parameters are in their definition domain, and to determine the gap between the current value and its domain. This characteristic allows quantifying the deviation from the norm. Two criteria

Table 1. Two different ways to express the safety time norm

	safety time definition domain
first expression	singleton, t_s
2nd expression	normal distribution, $\mu = t_s$, $\sigma^2 = 0.25$

can be used: firstly, the number of values of the behavior's parameters outside the limits defined in the norm; secondly, the gap between a generated value and its original specification.

For instance, consider the behavior of drivers regarding the safety distance on roads. In the Highway Code, only recommendations are provided: "allow at least a two-second gap between you and the vehicle in front on roads carrying faster-moving traffic and in tunnels where visibility is reduced" (rule 126 of the English Official Highway Code [12]). You can be fined for dangerous driving if you drive too close to the vehicle in front of you, but there is no obligation regarding this point. We define this as a norm, which can be expressed in different ways with our formalism (Table 1). Using the first expression, a behavior which instantiates this norm can take the value t_s, and belong to the norm. If it takes any other value $t = t_s + \delta,\ \delta \in [-t_s, +\infty]$ we observe a violation. We are also able to quantify the deviation: if $t_s = 2\,\text{s}$ and $\delta = 0.5\,\text{s}$, a deviation of 25 % is observed. This way, too deviant behaviors can be excluded. With the second expression, if $t_s = 2\,\text{s}$, a value of 1.5 s stays within the domain: no violation is observed. These two norms illustrate how norms definition can provide different permissiveness levels.

This quantification can be used to fulfill various needs. It allows us to exclude too deviant behaviors, as we are able to quantify the deviance and set limits on the potential gaps. It can also be used to create unexpected behaviors, even aberrant ones, and study their influence on the simulations.

3.3 Generating Behavioral Variety

The simulation is managed using a nondeterministic mechanism: global parameters describe the randomization of agents behavior. These parameters are used to generate every other randomized parameter in the simulation, and can themselves be randomly chosen.

With this mechanism, the randomization level of the simulation can be set. If it is defined at the simulation level, all structures are randomized using the

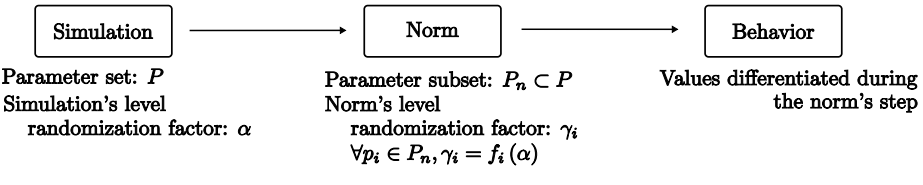


Fig. 4. Randomization mechanism allowing generating the behaviors

higher-level factor. However, if we decide to preserve more control on the agents characteristics, a different factor can be defined for each of them (Fig. 4). In addition, the degree of randomization of the simulation can also be chosen. The simulation can be either fully determined, with a simulation's level parameter set to 1, or totally randomized, with a simulation's level parameter set to 0. This generation mechanism is further detailed in [13].

4 Application to Traffic Simulation

One of the applications of this work is to reproduce various kinds of behaviors in traffic simulation. We present in the next section the first steps of the application of our model in the driving simulation software used at Renault, SCANeR™[1].

4.1 Driving Simulators and Traffic Simulation

Traffic simulation can be approached in several ways, depending on the requested level of detail. However, when dealing with a driving simulator, only a microscopic representation is suited: the vehicles driving around the interactive one should have a convincing behavior, which macroscopic models cannot provide. Driving simulators are used at Renault for different studies: ergonomics of the driver's cab, validation of embedded systems, comfort, design, validation of car lightings (Fig. 5)... The environment has thus to be as realistic as possible to allow the immersion of the users in the simulation and ensure results validity.

Various traffic management models have been developed in the driving simulators field during the last fifteen years. They use different decision models to simulate drivers behavior [14,15,16]. However, behavioral differentiation is not considered as a specific issue in these applications. In macroscopic simulations, this kind of mechanism sometimes exists for traffic generation functions [17].

Fig. 5. The dynamic simulator Ultimate at Renault *(left)*, and a screenshot of the SCANeR™ software with two visuals outputs, the traffic and the supervisory modules *(right)*

[1] SCANeR™ has initially been developed by Renault, and is now distributed and co-developed by Oktal (http://www.scaner2.com/).

4.2 Road Traffic Context

The normative system we place ourselves in is the road system. Various elements regulate it: the Highway Code first, which provides sets of rules, enforced by laws, and sets of recommendations; and second the habits established by drivers during their daily use of their vehicles.

The English Official Highway Code [12] explicitly presents a set of "must / must not" rules. They are associated to advices and recommendations, for which the code states that "although failure to comply with the other rules of the Code will not, in itself, cause a person to be prosecuted, The Highway Code may be used in evidence in any court proceedings under Traffic Acts to establish liability". Even if these additional rules are not subjected to automatic punishment, they are explicitly provided to establish a framework for the normative system. Other codes, like the French one, present the same kind of characteristics.

As for individual elements, several psychological factors are involved in drivers behavior [18]: personality, emotion, motivation and social behavior. Psychological based driver models have been developed [19], but the lack of links between measurable and psychological parameters makes their concrete application difficult. Indeed, drivers take into account various rules encountered in the real world [20]:

- formal rules (rules of the road),
- informal rules (practices or conventions which can be in contradiction with the formal rules, like not yielding at crossroads or roundabouts),
- design of the road (which is often the origin of informal rules appearance),
- and other drivers behavior (their current behavior as well as the anticipated one).

Driving presents several particularities: many rules are subject to interpretation, the road environment let people expose their personality, and the emotional state can influence the behavior. For instance, a driver may be dangerous even if he does not break any rule: over-cautious drivers interfering with the traffic flow can endanger others road users. The application of the rules can even differ from a country to another, or from a town to another, adding a dependence on environmental factors. A wide multiplicity of behaviors can be observed, which has to be reproduced in simulations to ensure the immersion of human drivers in the simulations.

4.3 Parameters of the Traffic Model

In traffic simulation, the individual characteristics of the agents are usually described by a set of numerical data used in the decision model. Among them acceleration, braking, security distance, security margin, or even psychological factors like time to collision or time to lane crossing are typically used.

In SCANeR™, the autonomous vehicles use a decision model based on a perception-decision-action architecture [11]. During the perception step, the driver identifies the elements it may interact with. It includes the roads,

the road signs, the other vehicles and the pedestrians. The decision step is built on three levels. First, a strategic level plans the itinerary. Then, a tactical level is applied to select the next maneuver to be executed: drive on, overtake, change lane, or stop. This step uses a finite state automaton, which transitions are sensible to different parameters. After the maneuver's choice, an operational level computes the resulting acceleration and wheel angles. Finally, the action model computes the next position, using a dynamic model of the vehicle.

Six different pseudo-psychological parameters are taken into account in this decision model:

- maximal speed: the maximal speed a driver will adopt,
- safety time: the security distance it will use, depending on its speed,
- overtaking risk: the risks a driver will accept to overtake, function of the available gaps with oncoming vehicles,
- speed limit risk: a factor allowing bypassing speed limits,
- observe priority and observe signs: boolean parameters regarding the respect of signalization and priorities.

Their values can be set without any consistency check, and no consistency of the resulting behavior is guaranteed.

4.4 Implementation

We chose in this work to apply the proposed differentiation model on the existing pseudo-psychological parameters. Indeed, they influence the behaviors of the drivers, and represent adapted inputs in the traffic model. This led to the institution whose parameters and associated values are presented in Table 2. For the purpose of our example, the institutional and environmental properties are defined as follows: we consider the institution is valid in right driving countries ($P_i = \{right\_driving\}$), and only in France and Italy ($P_e = \{France,\ Italy\}$).

Different norms can then be defined in the context of this institution. Two examples are presented in Table 3: normal and aggressive driving on highways. The norm *normal highway driving* uses all the parameters defined in the institution ($P_n = P$), the definition domains are restrictions of the institution ones. A driver applying this norm does not take risks to overtake, drive within the speed limits, do not bypass them, and observe both priorities and signalization. The second example represents the norm *aggressive highway driving*: again all parameters are used, but the definition domains are

Table 2. Institution with the existing parameters using the presented model

$P = \{p_i, i \in [1,6]\}$	$D_P = \{d_{p_i}, i \in [1,6]\}$
p_1 = maximal_speed	$d_{p_1} = [0, +\infty]$
p_2 = safety_time	$d_{p_2} = [0, +\infty]$
p_3 = overtaking_risk	$d_{p_3} = [0, 1]$
p_4 = speed_limit_risk	$d_{p_4} = [0, +\infty]$
p_5 = observe_signs	$d_{p_5} = \{true, false\}$
p_6 = observe_priority	$d_{p_6} = \{true, false\}$

Table 3. Norms describing normal and aggressive driving on a highway

	normal highway driving	aggressive highway driving
p_{n_1} = maximal_speed	$d_{p_{n_1}} = [100, 140]$	$d_{p_{a_1}} = [140, 160]$
p_{n_2} = safety_time	$d_{p_{n_2}} = [0.8, 5.0]$	$d_{p_{a_2}} = [0.1, 1.2]$
p_{n_3} = overtaking_risk	$d_{p_{n_3}} = [-0.5, 0.5]$	$d_{p_{a_3}} = [1.0, 2.0]$
p_{n_4} = speed_limit_risk	$d_{p_{n_4}} = [0.0, 1.0]$	$d_{p_{a_4}} = [1.0, 10.0]$
p_{n_5} = observe_signs	$d_{p_{n_5}} = \{true\}$	$d_{p_{a_5}} = \{true, false\}$
p_{n_6} = observe_priority	$d_{p_{n_6}} = \{true\}$	$d_{p_{a_6}} = \{true, false\}$

Table 4. A normal and a violating instantiation of the *aggressive highway driving* norm. Only one parameter, the *maximal speed*, is in violation.

	aggressive driver	violating aggressive driver
p_{b_1} = maximal_speed	$v_{p_{b_1}} = 150\,\text{km/h}$	$v_{p_{b_1}} = 210\,\text{km/h}$
p_{b_2} = safety_time	$v_{p_{b_2}} = 0.2\,\text{s}$	$v_{p_{b_2}} = 0.3\,\text{s}$
p_{b_3} = overtaking_risk	$v_{p_{b_3}} = 2.0$	$v_{p_{b_3}} = 1.8$
p_{b_4} = speed_limit_risk	$v_{p_{b_4}} = 1.6$	$v_{p_{b_4}} = 3.0$
p_{b_5} = observe_signs	$v_{p_{b_5}} = false$	$v_{p_{b_5}} = true$
p_{b_6} = observe_priority	$v_{p_{b_6}} = false$	$v_{p_{b_6}} = false$

adapted to reflect that aggressive driver take more risks, drive faster and use smaller security margins. The norm allows not respecting priorities or signalization. As for the properties sets, the institutional one remains unchanged, but the environmental properties now include a parameter to restrict its use to highways only, and to France only: $P_{norm_i} = P_{aggr_i} = \{right\_driving\}$ and $P_{norm_e} = P_{aggr_e} = \{France,\ highway\}$.

These norms allow generating various behaviors. In Table 4, two instantiation of the *aggressive highway driving* norm are presented. The first one does not violate the norm: every value remains in the definition domain defined by the norm. A driver using these parameters in the traffic model presents a consistent behavior, while showing aggressive characteristics, like following closely the vehicles in front of it. The second instantiation represents the kind of behavior that may be created when violations are allowed. Here, only one parameter has been generated outside the norms, the *maximal speed*. The generated value leads to a coherent behavior, but if the value had led to an inconsistent behavior (for instance 400 km/h), the behavior should have been excluded.

4.5 Experimental Results

To evaluate the improvements brought by the introduction of the normative model, simulations using different sets of norms were realized.

A database representing an 11 km long section of highway was used (Fig. 6). The vehicles were created at the beginning of the section, using a traffic demand of 3800 veh/h (1900 veh/h per lane on both lanes). The recording of traffic data

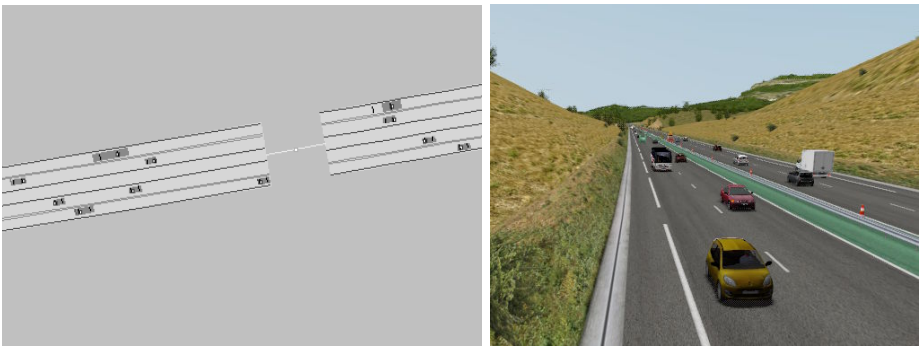


Fig. 6. 2D and 3D views of the highway database used for the experiment

was done using three detector placed on the highway, at kilometer 2.2, 6 and 10.8. The vehicles were created using the normative model, and the traffic model of the application then handled them during the simulation process. Three different sets of norms were used:

- no norms: all the vehicles are created with the same parameters,
- normal driver only: one norm is used, *normal highway driving*. Only one parameter is specified in the norm, the maximal speed. Its definition domain is a normal distribution of mean $\mu = 125$ and standard deviation $\sigma = 10$, truncated at 100 and 140 km/h,
- all norms: three norms are used, *cautious highway driving*, *normal highway driving* and *aggressive highway driving*. Each norm is defined with four parameters, which definition domains are truncated normal distributions. The values used are presented in table 5. The vehicles are created with the following proportions: 10 % cautious, 80 % normal, and 10 % aggressive.

Table 5. Cautious, normal and aggressive norms parameters

parameter	cautious	normal	aggressive
maximal speed	$[90, 125]$	$[100, 140]$	$[140, 160]$
	$\mu = 115$	$\mu = 125$	$\mu = 150$
	$\sigma = 10$	$\sigma = 10$	$\sigma = 5$
safety time	$[1.5, 5.0]$	$[0.8, 5.0]$	$[0.1, 1.2]$
	$\mu = 2.0$	$\mu = 1.5$	$\mu = 0.8$
	$\sigma = 0.5$	$\sigma = 0.5$	$\sigma = 0.4$
overtaking risk	$[-0.5, 0.5]$	$[-0.5, 0.5]$	$[1.0, 2.0]$
	$\mu = 0.0$	$\mu = 0.0$	$\mu = 1.5$
	$\sigma = 0.25$	$\sigma = 0.25$	$\sigma = 0.5$
speed limit risk	$[0.0, 1.1]$	$[0.0, 1.1]$	$[1.0, 10.0]$
	$\mu = 1.0$	$\mu = 1.0$	$\mu = 1.5$
	$\sigma = 0.05$	$\sigma = 0.05$	$\sigma = 0.25$

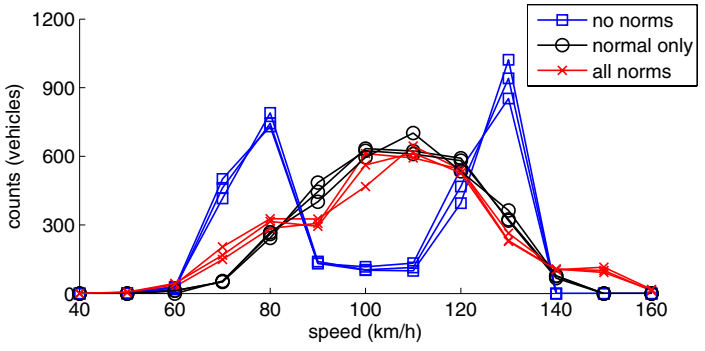


Fig. 7. Distribution of vehicles speeds at kilometer 6

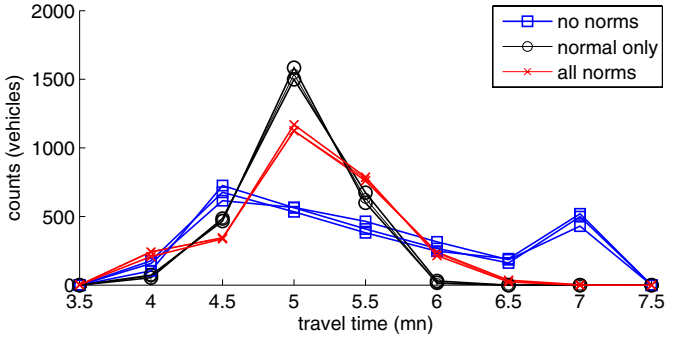


Fig. 8. Total travel time on the database

For each set of norms, the data obtained during three different runs are presented (Figs. 7 and 8), the duration of each simulation being one hour. The Figure 7 represents the distribution of vehicles speeds at the kilometer 6 (second detector). The first case, where no norm is used, shows a concentration of the speeds in two main areas: between 70 and 90 km/h for 46 % of the vehicles, and around 130 km/h for 40 % of them. This distribution is explained by the parameters similarity: vehicles are not able to take advantage of the traffic flow variations, which results in a slow right lane and a fast left one with few lane changes. In the second case, with one norm, 60 % of the vehicles speeds are between 90 and 115 km/h, and 30 % between 115 and 140 km/h. The resulting distribution is more balanced, but the average speed remains quite low, at 100.4 km/h. The last case, with three norms, presents a wider distribution of the speeds, and a slightly increased average speed (103.7 km/h).

The total travel time for each vehicle is presented in Figure 8, and provides interesting complementary results. As for the speeds, the use of norms produces more balanced distributions of results. In addition, the distributions widen when the number of used norms increases: a higher variety of behaviors results in more differentiated travel times. When studying the average travel times of the vehicles

Table 6. Average travel times

	avg travel time	evolution
no norm	5 mn 35 s	+13.2 %
normal only	4 mn 56 s	ref.
all norms	5 mn 14 s	+6.0 %

through the whole section (Table 6), we can also note that even if the average speed increases slightly when using more norms (+3.3 %), the travel time do not decrease, but increase (+6 %). The variety of behaviors explains again this result: more speeding vehicles are present, but the dynamicity of the traffic limits their progression.

However, different elements concerning the experiment have to be discussed. First, the norms choice, and the values used in the norms. The norms were chosen to reflect classifications defined in driving psychology. They may change to include more variety, or according to the population studied. As for the values used, they have been chosen empirically. An important improvement would be using calibration with real data, which is currently under work.

Second, the use of statistical data hides some of the characteristics of the traffic. Even if some properties appear, the visual observation of the traffic flows shows other particularities: increasing the variety increases highly the variety of individual behaviors in the traffic (overtakings, speed choices...). These results do not appear in the measured data, and we need to introduce new indicators allowing illustrating and quantifying these elements.

Finally, the possibility to generate violating behaviors was not exploited in these simulations. Even when creating aggressive drivers, we remained in the limits of the corresponding norm. This point will be introduced in further experiments, to simulate for instance erratic behaviors in the traffic.

5 Conclusion

In this paper we have presented an approach to model behavioral differentiation as deviations from the norm in simulations of spatially situated agents. Such behavioral variety is needed in microscopic simulations, where it is an important realism criterion. The institutional environment is composed of an institution, norms and behaviors. The institution manages a set of parameters associated to their definition domains. The norms are subsets of these parameters and domains, and behaviors are instantiations of a norm. The values of behaviors parameters can be in or outside the definition domain provided by the norm. With this model, any kind of behavior can be generated, either matching or violating the specified norms. We are also able to quantify the deviance rate of these behaviors. Finally, this approach has been applied to traffic simulation. In this first step, the existing parameters of the traffic model have been used to generate various agents behaviors. Statistical experimental results showed

that the introduction of different norms improves the behavioral variety in the simulation, while allowing controlling the consistency of the behaviors.

References

1. Boella, G., van der Torre, L.: An architecture of a normative system: count-as conditionals, obligations and permissions. In: Int. Joint Conf. AAMAS, New-York, USA, pp. 229–231 (2006)
2. Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: Knowledge Representation, Whistler, Canada, pp. 255–265 (2004)
3. Savarimuthu, B., Cranefield, S., Purvis, M., Purvis, M.: Role model based mechanism for norm emergence in artificial agent societies. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 203–217. Springer, Heidelberg (2008)
4. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11, 307–360 (2005)
5. Noriega, P.: Agent mediated auctions: The Fishmarket Metaphor. PhD thesis, Universitat Autonoma de Barcelona (1997)
6. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS, vol. 2333, pp. 348–366. Springer, Heidelberg (2002)
7. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Towards self-configuration in autonomic electronic institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 229–244. Springer, Heidelberg (2007)
8. Doniec, A., Espié, S., Mandiau, R., Piechowiak, S.: Non-normative behaviour in multi-agent system: Some experiments in traffic simulation. In: IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, Hong Kong, China, pp. 30–36 (2006)
9. Castelfranchi, C., Dignum, F., Jonker, C.M., Treur, J.: Deliberative normative agents: Principles and architecture. In: Jennings, N., Lesperance, Y. (eds.) Agent Theories Architectures and Languages, Orlando, USA, pp. 206–220 (1999)
10. Lacroix, B., Mathieu, P., Picault, S.: Time and space management in crowd simulations. In: European Simulation and Modelling Conference, Toulouse, France, pp. 315–320 (2006)
11. Lacroix, B., Mathieu, P., Rouelle, V., Chaplier, J., Gallée, G., Kemeny, A.: Towards traffic generation with individual driver behavior model based vehicles. In: Driving Simulation Conference, Iowa City, USA, pp. 144–154 (2007)
12. Department for Transport: The Official Highway Code. The Stationery Office (2007)
13. Lacroix, B., Mathieu, P., Kemeny, A.: A normative model for behavioral differentiation. In: IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, Sydney, Australia, pp. 96–99 (2008)
14. Espié, S., Saad, F., Schnetzler, B., Bourlier, F., Djemane, N.: Microscopic traffic simulation and driver behaviour modelling: the Archisim project. In: Road Safety in Europe and Strategic Highway Research Program, pp. 22–31 (1994)
15. Wang, H., Kearney, J., Cremer, J., Willemsen, P.: Steering behaviors for autonomous vehicles in virtual environments. In: IEEE Virtual Reality Conference, pp. 155–162 (2005)

16. Olstam, J.: Simulation of rural road traffic for driving simulators. In: Transportation Research Board, Washigton D.C., USA (2005)
17. Barcelo, J., Casas, J.: Dynamic network simulation with AIMSUN. In: International Symposium on Transport Simulation, Yokohama, Japan (2002)
18. Dewar, R.E.: Individual differences. In: Dewar, R., Olson, P. (eds.) Human Factors in Traffic Safety, pp. 111–142 (2002)
19. Keskinen, E., Hatakka, M., Laapotti, S., Katila, A., Peraaho, M.: Driver behaviour as a hierarchical system. In: Rothengatter, T., Huguenin, R.D. (eds.) Traffic and Transport Psychology, pp. 9–29. Elsevier, Amsterdam (2004)
20. Björklung, G.M., Aberg, L.: Driver behaviour in intersections: Formal and informal traffic rules. Transportation Research Part F 8, 239–253 (2005)

Categorizing Social Norms in a Simulated Resource Gathering Society

Daniel Villatoro and Jordi Sabater-Mir

Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Bellatera, Barcelona, Spain
{dvillatoro,jsabater}@iiia.csic.es

Abstract. Our main interest research is focused on reaching a decentralized form of social order through the usage of social norms in virtual communities. In this paper, we analyze the effects of different sets of social norms within a society. The simulation scenario used for the experiments is a metaphor of a resource-gatherer prehistoric society. Finally, we obtain a qualitative ranking of all the possible sets of social norms in our scenario performing agent-based simulation.

1 Introduction and Related Work

Social norms are part of our everyday life. They help people self-organizing in many situations where having an authority representative is not feasible. On the contrary to institutional rules, the responsibility to enforce social norms is not the task of a central authority but a task of each member of the society. From the book of Bicchieri [1], the following definition of social norms is extracted: "The social norms I am talking about are not the formal, prescriptive or proscriptive rules designed, imposed, and enforced by an exogenous authority through the administration of selective incentives. I rather discuss informal norms that emerge through the decentralized interaction of agents within a collective and are not imposed or designed by an authority". Social norms are used in human societies as a mechanism to improve the behaviour of the individuals in those societies without relying on a centralized and omnipresent authority. In recent years, the use of these kinds of norms has been considered also as a mechanism to regulate virtual societies and specifically societies formed by artificial agents ([2,3,4,5]). From another point of view, the possibility of performing agent based simulation on social norms helps us to understand better how they work in human societies.

One of the main topics of research regarding the use of social norms in virtual societies is how they emerge, that is, how social norms are created at first instance. This has been studied by several authors ([6,7,8,9,10,11]) who propose different factors that can influence this emergence. We divide the emergence of norms in two different stages: (a) how norms appear in the mind of one or several individuals and (b) how these new norms are spread over the society until they become accepted social norms. We are interested in studying the second

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 235–249, 2009.

stage, the spreading and acceptance of social norms, what Axelrod [6] calls *norm support*. Our understanding of norm support deals with the problem of which norm is established as the dominant when more than one norm exists for the same situation. In the literature we can find several works ([7,9]) that address with this problem, using a prisoner's dilemma as evaluation function, converting the problem of norm support in a coordination problem, where the agents have to learn to cooperate with the rest of the society, otherwise any kind of social punishment will be applied to them.

Our model, in contrast to those solving coordination problems, can deal with social norms that are not representable in a decision table and the rewards for following a certain norm are not known a priori. A similar approach can be found in the work of Cecconi and Parisi [12], where they also deal with a simulated resource consuming society. In their work, agents do not know beforehand how good the sets of social norms they follow are, even though the authors only consider two well differentiated sets of social norms (individual strategy or collective strategy of resource consumption). However, a society can have several (more than just two as we have already seen in the literature) sets of social norms abided by different members of the society. In the work of Sen [13], we observe that the authors present 6 different strategies (or sets of social norms), but they study the behaviour of mixed populations of these kinds of agents. Nevertheless, each of these sets of social norms, acting individually, can be of different quality with respect the society's goal. Therefore, it is useful to know beforehand the quality of a set of norms in a society, assuming that all the agents share the same set of social norms. In this paper we present a deep analysis of simulation results and the statistical techniques used *to establish a ranking of quality of all the possible sets of social norms* that members of a well-defined society can abide by. The assumption addopted is that all the members share the same set of social norms, with the hypothesis that, when agents find themselves in a socially mixed society, they will tend to a common set of norms, and such set of norms should be optimal. The research contained herein follows that performed by [14] where a genetic algorithm was the mechanism in charge of finding the most efficient set of norms in a given society. The main motivation (and part of future work) of this research is, once the quality of each different set of social norms is defined, to create simulations of heterogeneous societies. In these simulations agents will be loaded with different sets of social norms, and agents will be provided with the ability of changing their set of social norms. Therefore, we plan to observe a convergence of all the agents into a set of social norms. Our final goal is to study the mechanisms that favour that the final dominant set of social norms is the best in the ranking we have previously established. The article is structured as follows: firstly, we present the motivation of the problem and the inspiration we are using for the simulation scenario. Secondly, it is described the problem we deal with in this article, as well as the hipothesis. Subsequently all the details of the simulation model are specified. Thirdly, the experimental setting is introduced and the results of the experiments are analyzed. Finally, we draw some conclusions from the results obtained.

2 Reference Scenario

In order to design an scenario where the usage of social norms is significant, we are inspired by real life examples ([15], [16]), where the usage of social norms is vital for the survival of the society. The society we use for our experiments is a resource-gatherer distributed and decentralized society. All the members of the society survive by consuming resources that appear randomly in the environment and exchanging the resources among them by *abiding to a set of social norms*. Depending on the quality of these social norms, the society succeeds in the task of increasing the average life expectancy of its members.

The application domain of this research is directly related to an ongoing research which is carried out by a group of archaeologists. We are presented a non-prehistoric society, already extinguished, known as *'the Yámanas'*. This society was located in Southern Argentina and are one of the groups of the societies commonly known as 'canoeros'. They lived there for around 6000 years in a very hostile environment. The main success, and reason of study, of this peculiar society is their ability of auto-organization: the *Yámanas* were able to auto-organize themselves as a hunter-gatherer society. The archaeologists consider as the hypothesis that the key of success in this society was due to their strong respect for a known set of social norms (represented as a set of myths). These social norms regulated, amongst other behaviours, the resource exchange between the *Yámanas*. From the study of Gusinde [17], we extract that social norms for resource exchange regulation only made sense in such societies when the resources to be exchanged would appear sporadically although of a large contribution when they appear (e.g. finding a whale on the beach was a huge amount of resources but it would not happen frequently). Therefore, we adapt the parameters of the simulation to this scenario.

We want to stress that even though we inspired our simulations by the previously described society, the simulation scenario is a simplification of it. Consequently, we do not intend to affirm that the results obtained out of our simulations, as they are now, are directly applicable to real societies. Notwithstanding, the results have relevance for societies of virtual agents.

3 Statement of the Problem

The problem to be faced in the following sections is a study of the effects of each set of social norms within the society that uses them. We perform an exhaustive analysis of every possible set of social norms in our resource-gatherer society, forcing each time all the members to share the same set of social norms. This analysis provides us with the necessary information to *establish a classification of sets of social norms depending on their quality*. The quality measure used in our experiments is the Average Life Expectancy of the agents. Having fixed the ranking , we observe the characteristics that make a set of social norms optimal, with the intention of applying this characteristics to different scenarios in the future work. Our hypotheses are:

- **H1** - Different sets of social norms obtain different results on the quality measure we are using.
- **H2** - Environmental settings can affect the ranking of social norms.
- **H3** - Social norms promoting selfishness generate heterogeneous societies (as dictatorships).
- **H4** - Homogeneous societies are obtained with sets of social norms that promote altruism.

4 Simulation Model

We use a multi-agent system for our simulation. This multi-agent system is defined as an undirected graph: $MAS = \langle A, Rel \rangle$, where $\mathcal{A} = \{Ag_1, Ag_2, Ag_3, \ldots, Ag_n\}$ is a set of n agents representing the vertices of the graph, with $n \geq 1$; and Rel the set of relations (edges) between the agents. All the neighbours of distance 1 in the graph MAS of a certain agent is defined as the *neighbours network* of this agent. All the agents are initially loaded with 100 resource units. The simulation algorithm is based on a discrete step timing model, where each time step the algorithm observes the state and consequent actions of each agent before ticking another time step. Every time step, the simulation algorithm runs over every agent. The order in which the algorithm runs over the agents is randomly changed each time step. In this way all the agents are able to execute their actions, in a random order each time step, anulling any kind of advantage of one agent over the rest.

Each agent consumes one resource unit each time step as energy consumption for survival. When one agent exhausts its resources, it dies. After dying, agents are able to ressurrect with the initial resource conditions, after recalculating its *Average Life Expectancy* (ALE). This ALE is calculated by averaging the age of death plus the previous ALE. At the beginning of the simulation, all agents are loaded with an initial ALE of 100.

Firstly, in each time step, our algorithm evaluates (following continuous uniform probability distibution) if each of the agents have to find resources by observing the agent *Resource Gathering Probability*, that is defined as:

Resource Gathering Probability (P_{rg}) is ranked in the interval $[0, 1]$. P_{rg} specifies the probability an agent has to find resources each time step. In case the algorithm evaluates that an agent has to find resources, the agent will receive a large amount of resources that can either use for its own consumption or for donating.

Secondly, in each time step, our algorithm evaluates if an agent has to meet another agent by observing the agent *Interaction Probability*, that is defined as: *Interaction Probability (P_{int})* is ranked in the interval $[0, 1]$. P_{int} specifies the probability of an agent to meet another agent connected to it.

In case the algorithm evaluates positively that an agent has to meet another one, it randomly chooses another agent from the agent's neighbours network. The interactions among agents are done always in pairs, and both agents have to choose an action when interacting. This decision is taken following the *set of social norms* that each agent has internalized. The set of norms specifies if the agent has to give or not give resources to the other agent, depending on both

Table 1. Situations and Actions. Structure of a set of social norms.

Situation		Action
Starving(Me)	Starving(You)	To Give / Not To Give
Starving(Me)	Plenty(You)	To Give / Not To Give
Starving(Me)	Normal(You)	To Give / Not To Give
Plenty(Me)	Starving(You)	To Give / Not To Give
Plenty(Me)	Plenty(You)	To Give / Not To Give
Plenty(Me)	Normal(You)	To Give / Not To Give
Normal(Me)	Starving(You)	To Give / Not To Give
Normal(Me)	Plenty(You)	To Give / Not To Give
Normal(Me)	Normal(You)	To Give / Not To Give

agent's resource levels. In order to formalize our concept of social norm, we first need to define several terms.

All agents can perceive a finite set of *observables* $\mathcal{O}$, and each element of the set is denoted as *ob*. Every agent also has a finite set of *actions* A, and each element of the set is denoted as *a*.

Every agent can find itself in a finite set of different *situations* $\mathcal{S}$, and each element of the set is denoted as $sit \subset \mathcal{O}$. In other words, a *situation* is a combination of different observables.

Given that, a **social norm** SN_i is a tuple formed by a situation and an action: $SN_i = \{\langle sit_g, a_h \rangle \mid sit_g \in \mathcal{S}, a_h \in A\}$.

In our scenario, the set of observables is formed by the following propositional terms: $\mathcal{O} = \{$ *Plenty(Me)*, *Plenty(You)*, *Normal(Me)*, *Normal(You)*, *Starving(Me)*, *Starving(You)*$\}$, where: *Plenty(X)* indicates that *Agent's X* resource level is over 100 units; *Normal(X)* indicates that *Agent's X* resource level is between 25 and 100 units; and, *Starving(X)* indicates that *Agent's X* resource level is below 25 units. The values that X can take are *Me* and *You*, representing the acting agent and the partner agent in the interaction. When two agents meet, each agent is able to observe its own level of resources and its opponent level. The whole list of possible situations (formed by two observables) in which an agent may find itself can be seen in Table 1. The set of possible actions are $A = \{\text{Give Resources}, \text{Do not Give Resources}\}$. The combination of all possible situations associated to an action generates a *set of social norms*.

Each agent always abides by the set of social norms that it has internalized. When the social norm indicates to give resources, the agent has to decide the amount of resources it gives. Each agent has been provided with a *Donation Reasoning Process* that allows it to calculate the amount of resources to donate. The Donation Reasoning Process is the following:

if $(Age_A \geq ALE_A)$ *and* $(Resources_A \geq PlentyLevel)$ **then**
  $Donation = SharingFactor \times (Resources_A - PlentyLevel)$
else
  $Donation = (1 - SharingFactor)^2 \times Resources_A$
end

Age_A corresponds to how old Agent A is. ALE_A refers to the Average Life Expectancy of Agent A. $Resources_A$ is the amount of resources that Agent A posses at that moment. *PlentyLevel* is the level in which the agent is considered to be plenty. And *SharingFactor* is a factor applied to donate a relative amount of the total. In the experiments studied herein this sharing factor is fixed on a 70%. In other words, when an agent has more resources than what it needs to increase its average life expectancy, it donates more; when an agent do not have enough resources, it donates a smaller amount. The donation reasoning process has been designed in such a way so that it fulfils the motivation of the scenario we are simulating that were introduced in previous sections.

5 Experiments and Results

Once the characteristics of the simulation platform have been grounded and the architecture of the agents is clear, we make use of them to test our theories of how efficient a set of social norms can be.

We suspect that depending on the amount of resources available in the environment, a different set of social norms will be the most efficient in every scenario, changing therefore the behaviour of the agents depending on the availability of resources.

5.1 Experiment Design

We need to test every single set of social norms over a society where every member of the society shares the same set of social norms. We have decided to load into the simulation a society with the following characteristics:

- The number of agents loaded in the simulation has been fixed to 90. This amount of agents allow us to approximate the society result to a normal distribution, so that it fulfills the central limit theorem[1].
- Fully Connected Neighbour Network: every agent is connected to all the other agents in its neighbour network.
- All the agents have the same Interaction Probability, and it has been fixed to $P_I(Agent_i) = 0.1$. This parameter is fixed to this value to avoid the continuous interactions among agents. A limited number of interactions makes the result of this interaction more important when happening. [2]
- All the agents have the same Resource Gathering Probability, and this parameter ($P_{RG}(Agent_i)$) is variable depending on the experiment.
- All the agents have the same set of social norms. Every possible set of social norms is loaded into the agents, executed and analyzed its effect after a period of time.
- When agents find resources, 250 units of resources are found. [2]

[1] The central limit theorem states that the re-averaged sum of a sufficiently large number of identically distributed independent random variables each with finite mean and variance will be approximately normally distributed [18].

[2] This value has been chosen to fulfil the reference scenario previously presented and obtained from [17].

Apart from these parameters, we also have to specify the simulations parameters. All simulations are run for 250000 steps. In each simulation, a different set of social norms is loaded, until all possible sets of social norms have been executed. For each different set of norms, 20 simulations are run and certain parameters are saved. These parameters are: Average Life Expectancy of each agent, Standard Deviation of the Average Life Expectancies of the society, and Median Average Life Expectancy of the society.

As it was explained in Section *Simulation Model*, each agent could find itself in 9 possible different situations. In each of these situations, an agent always has two options: to give or not to give resources. Therefore, 2 *actions* raised to the power of 9 *situations* gives us a result of 512 different sets of social norms that will be studied separately.

5.2 Experiment 1

In this first experiment we have fixed $P_{RG}(Agent_i) = 0.0025$. This value indicates, for example, that in a grid world of 100 cells in each side, 25 cells (out of 10000) would be loaded with resources every time step. We consider it a *low* resource gathering probability, which do not allow the society to perpetuate. Therefore, we are interested in finding out which are the sets of norms that lengthen the average life expectancy of the society. After running an exhaustive test over all the possible set of social norms, we can observe the results in the following figure. The horizontal axis represents each one of the 512 possible sets of social norms. The vertical axis represents the mean of the median average life expectancy of the society from each of the 20 simulations.

H1 - Different sets of social norms obtain different results on the average life expectancy of the agents is verified with the results. In same environmental conditions, different sets of social norms produce different results in the agents

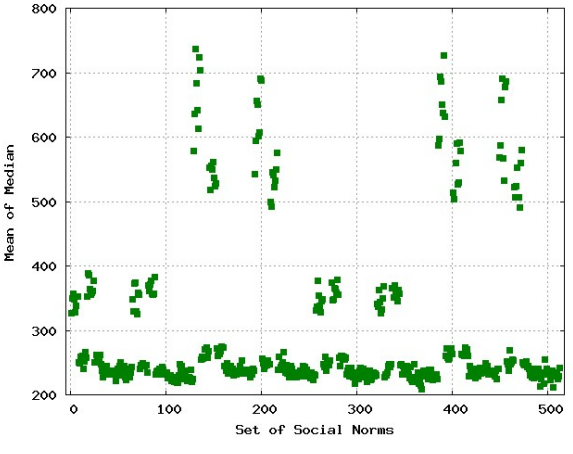


Fig. 1. Median Average Life Expectancy using different sets of social norms. $P_{RG}(Agent_i) = 0.0025$.

average life expectancy. The society, notwithstanding the social norms used, does not get to perpetuate for the whole simulation in any of the simulations. Therefore we observe which sets of norms obtain the best results. In Figure 1, we can perfectly distinguish between three different levels:

1. In the first level (median average life expectancy (ALE) lower than 300) we define the *Bad* sets of social norms.
2. In the second level (median ALE between 300 and 400) we define the *Average* sets of social norms.
3. In the third level (median ALE higher than 400) we define the *Good* sets of social norms.

In Figure 1, and in the levels aforementioned, we constantly refer to the mean of the median ALE. This median ALE represents information from only one member of the society, and does not provide us a with precise idea of how the rest of the society has behaved. It could happen that in two different societies with the same median ALE, the distance between the best and the worst member of the society was very different: one very large, representing a heterogenous society; and one very small, representing a homogenous society. In order to observe the homogeneity of each society, produced by the sets of social norms, we observe also the Average Standard Deviation of the simulations. If the Average Standard Deviation is low, this shall mean that all the agents have obtained similar results, obtaining consequently, an homogeneous society.

In Figure 2, we can observe four different data clusters:

– The lowest one (A) indicates a poor performance of these sets of social norms that this cluster holds. Although the bad performance of the set of norms in respect to the median average life expectancy of the society, it shows a very low standard deviation. The average median life of the agents is relatively

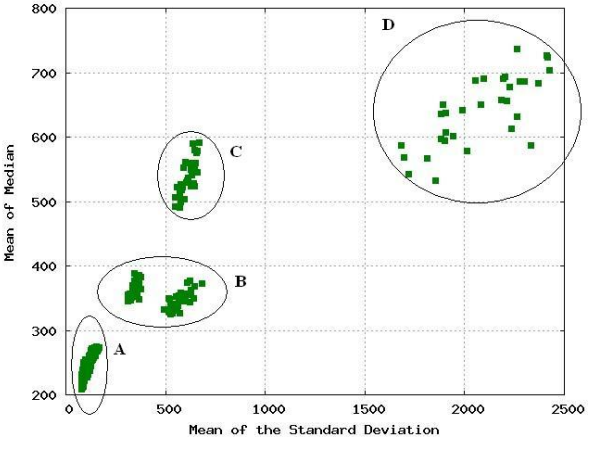


Fig. 2. Median Average Life Expectancy VS Mean of Standard Deviation. $P_{RG}(Agent_i) = 0.0025$.

low, but, so it is the standard deviation, which means that all the agents inside these societies obtain similar ALEs. The sets of norms in this cluster are tagged as *low*.

- The following one (B) shows an average performance. Inside this cluster it can be seen two smaller ones. One of the smaller clusters represents more homogeneous (referring tothe resulting population) sets of norms than the other one, although the median life of the agents is average with respect to the rest of the social norms. The sets of norms in this cluster are *medium*.
- The third cluster (C) shows the sets of social norms that we define as *high*. Societies using these sets of norms obtain a good median ALE, similar (slightly smaller in this third cluster) to the best cluster. It also results in a more homogeneous society than the last one. The sets of norms in this cluster can be tagged as *high and homogeneous*.
- The last cluster (D) is the most dispersed one. Although the performance in the "Mean of Median" axis is the highest, it is also the cluster that shows a higher standard deviation. These sets produce societies in which the "median agent" obtains a very good ALE, although the rest of agents obtain very different values. Therefore we can state that the sets of norms in this cluster are *high but heterogeneous*.

The sets of norms that show a good (high) performance deserve a deeper study. Consequently we extract such sets of norms and analyze the characteristics of both high clusters (C and D).

The sets of norms obtained in the heterogeneous cluster are the ones with the following IDs: 128 - 135, 192 - 199, 384 - 391, 448 - 455.

Each of the sets of social norms corresponds to a complete table of situations and its corresponding action. For example, the sets of norms identified as 128 - 135 are represented in Fig. 3. In each of the columns we can identify the action that is associated to the corresponding situation: To Give Resources (G) or Not To Give Resources (N G).

The set of norms in Fig. 3 (128-135) can be simplified into a more generalized one. This generalization is done following the theories of Karnaugh maps. By observing the three middle rows, these correspond to the situations *Plenty(Me)* and all the three possible observables for *You*. Therefore, and pursuant to the theory of Karnaugh maps, we generalize that the corresponding action for the situations with the observable *Plenty(Me)* is always *Do not give*, without considering the *You* observables (regardless of the value thay may hold, result would

Situation		Set 128	Set 129	Set 130	Set 131	Set 132	Set 133	Set 134	Set 135
Starving(Me)	Starving(You)	N G	G	N G	G	N G	G	N G	G
Starving(Me)	Plenty(You)	N G	N G	G	G	N G	N G	G	G
Starving(Me)	Normal(You)	N G	N G	N G	N G	G	G	G	G
Plenty(Me)	Starving(You)	N G	N G	N G	N G	N G	N G	N G	N G
Plenty(Me)	Plenty(You)	N G	N G	N G	N G	N G	N G	N G	N G
Plenty(Me)	Normal(You)	N G	N G	N G	N G	N G	N G	N G	N G
Normal(Me)	Starving(You)	N G	N G	N G	N G	N G	N G	N G	N G
Normal(Me)	Plenty(You)	G	G	G	G	G	G	G	G
Normal(Me)	Normal(You)	N G	N G	N G	N G	N G	N G	N G	N G

Fig. 3. Sets of Norms 128-135

not vary). In a similar way we generalize the last three rows, corresponding to the situations with the observable *Normal(Me)*. Finally, the first three rows, corresponding to the situations with the observable *Starving(Me)*, can be omitted. This is also done following Karnaugh maps theory. Since all possible combinations are covered, we can consider that that situation is not meaningful when extracting the generalization. The resulting generalization is:

If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$
If $Normal(Agent_A)$
If$Plenty(Agent_B)$ **Then** Give Resources to $Agent_B$
Else Do Not give Resources to $Agent_B$

By repeating the previous generalization procedure with the rest of sets of social norms, we obtain the following ("abstracted") sets of social norms:

1. **For the Sets of Norms (128-135):**
If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$
If $Normal(Agent_A)$
If$Plenty(Agent_B)$ **Then** Give Resources to $Agent_B$
Else Do Not give Resources to $Agent_B$

2. **For the Sets of Norms (192-199):**
If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$
If $Normal(Agent_A)$
If ($Plenty(Agent_B)$ **or** $Starving(Agent_B)$) **Then** Give Resources to $Agent_B$
Else Do Not give Resources to $Agent_B$

3. **For the Sets of Norms (384-391):**
If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$
If $Normal(Agent_A)$
If ($Plenty(Agent_B)$ **or** $Normal(Agent_B)$) **Then** Give Resources to $Agent_B$
Else Do Not give Resources to $Agent_B$

4. **For the Sets of Norms (448-455):**
If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$
If $Normal(Agent_A)$ **Then** Give Resources to $Agent_B$

Moreover, the generalization process can be performed on these resulting four generalized sets of social norms, obtaining just the last of the generalized set of social norms, since this one represents the most general situation. One conclusion that we may extract from this experiment is: when being an agent in resource-scarce environments, do not consider the others state, give only when you are normal and do not give when you are plenty of resources. This kind of norms promote the enrichment of those who are *Plenty*, favouring from those that continously die and ressurect, and not returning anything to the society. Thus, we have obtained a selfish society, but remembering that obtains good results although in an heterogeneous manner. Therefore, *H3 - Social norms promoting selfishness generate heterogeneous societies* is confirmed.

We still have to analyze the homogeneous cluster. The norms extracted (following the same previous procedure) from the homogeneous-high cluster are the following:

1. **If** ($Plenty(Agent_A)$ **or** $Normal(Agent_A)$)
If $Plenty(Agent_B)$ **Then** Give Resources to $Agent_B$
Else Do Not give Resources to $Agent_B$

2. **If** $Normal(Agent_A)$
   If ($Plenty(Agent_B)$ **or** $Starving(Agent_B)$) **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$
   If $Plenty(Agent_B)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$

3. **If** $Normal(Agent_A)$
   If ($Plenty(Agent_B)$ **or** $Normal(Agent_B)$) **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$
   If $Plenty(Agent_B)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$

4. **If** $Normal(Agent_A)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$
   If $Normal(Agent_B)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$

On the other hand, these norms, in contrast to the heterogeneous norms, do pay attention on the other agents state to decide the action to take, confirming that *H4 - Homogeneous societies are obtained with sets of social norms that promote altruism.* Possibly, this refinement in the decision process is the cause of the homogeneity.

5.3 Experiment 2

In this second experiment we have increased the amount of resources by fixing $P_{RG}(Agent_i) = 0.004$. We consider it a probability where agents, depending on the efficiency of the set of social norms, can achieve a good performance. Therefore, in this experiment we pursue the same objective described in Experiment 1: to find which are the codes that lengthen the average life expectancy of the

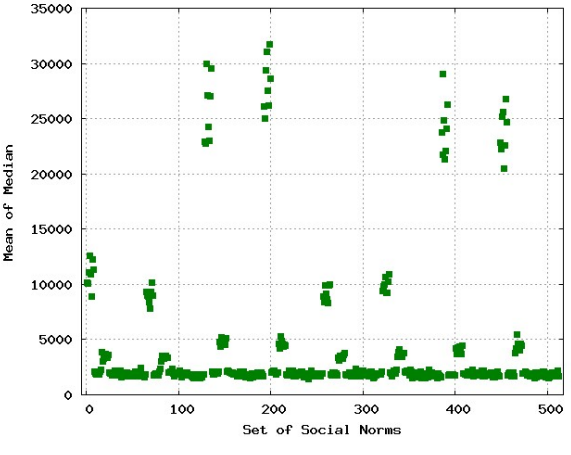


Fig. 4. Median Average Life Expectancy using different sets of social norms. $P_{RG}(Agent_i) = 0.004$.

society. After running an exhaustive test over all the possible set of social norms, we observe the results showed in Figure 4.

In Figure 4, we can observe a similar pattern of the distribution of the results over the space search. Although the scale in the axis of mean of median is larger this time, we can observe three levels as well:

1. In the first level (median ALE lower than 6000), we identify the *Bad* sets of social norms.
2. In the second level (median ALE between 6000 and 14000), we identify the *Average* sets of social norms.
3. In the third level (median ALE higher than 14000), we identify the *Good* sets of social norms.

At this time we also study the results in terms of homogeneity. This can be observed in the following figure.

As it happened in the first experiment, in Figure 5 we can observe four different data clusters. This time, it is more difficult to affirm which of them is the best cluster with respect to the others. On the one hand we have sets (A and B) that obtain poor results on the "mean of median" scale, but with a very low standard deviation. On the other hand, we have the most dispersed cluster (D), which obtains the best results, although showing a very high standard deviation. Finally, the third cluster (C), which obtains lower results than the fourth one, despite also having a lower standard deviation. However, when compared to the second cluster, we can observe a significant raise in the standard deviation for a not much significant raise in the "mean of median" scale. Accordingly, a decision has to be taken; sets of norms that produce: either the wealthiest society but with a high heterogeneity, or, a wealthy society (but not as wealthy as the previous one) but with a lower heterogeneity too.

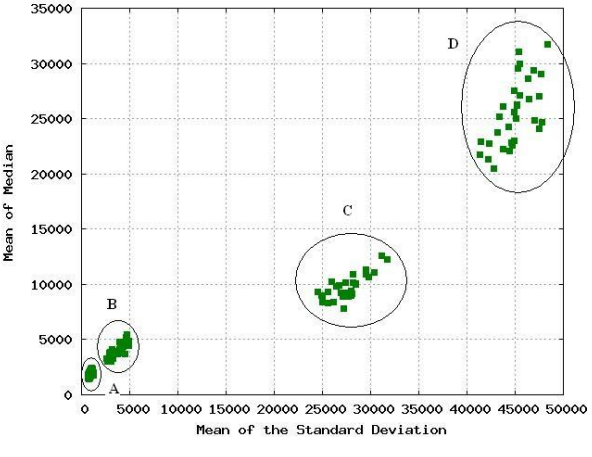


Fig. 5. Median Average Life Expectancy VS Mean of Standard Deviation. $P_{RG}(Agent_i) = 0.004$.

Despite this discussion, we would also like to observe the norms producing the two highest clusters that previously we distinguished between homogeneous and heterogeneous.

The sets of norms obtained in the heterogeneous cluster are exactly the same that the ones obtained in the first experiment.

The sets of norms obtained in the homogeneous cluster are:

1. **If** ($Plenty(Agent_A)$ **or** $Normal(Agent_A)$) **Then** Do Not give Resources to $Agent_B$

2. **If** $Normal(Agent_A)$
   If $Starving(Agent_B)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$

3. **If** $Normal(Agent_A)$
   If $Normal(Agent_B)$ **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$

4. **If** $Normal(Agent_A)$
   If ($Starving(Agent_B)$ **or** $Normal(Agent_B)$) **Then** Give Resources to $Agent_B$
   Else Do Not give Resources to $Agent_B$
   If $Plenty(Agent_A)$ **Then** Do Not give Resources to $Agent_B$

These norms are slightly different from those obtained in the first experiment. In these sets of norms, the *Starving* agents might still get some resources from other agents, while in the other example did not happen. These favouring to the *Starving* agents is due to the amount of resources; in this scenario is easier for the agents to find resources, therefore, makes sense to help them all. These differences confirm *H2 - Environmental settings can affect the ranking of social norms*. All the sets can be summarized into the last one. In these sets of norms we can still confirm the theory proposed at the end of the first experiment: to obtain a homogeneous society agents still have to pay attention on the other agents state to succeed.

6 Conclusions and Future Work

We have presented in this article a simulated society and an exhaustive study of social norms oriented to share resources that members of such society might use. From this analysis, we are now able to establish a quality scale of the different sets of social norms when acting separately.

We can conclude that selfish behaviours promote the proliferation of dictatorships of resources (some agents holding the majority of resources without sharing them with the rest of the society), consequently obtaining an heterogeneous society. On the contrary, in order to obtain homogeneous societies, the sets of norms have to promote altruism (making agents share resources in an intelligent way).

In this article, we have assumed that all members of the society share the same set of social norms. This assumption cannot be made when trying to simulate a real-life environment where to apply social norms as it could be a peer-to-peer

information market. In this kind of real problems, it might happen that each individual uses a different set of social norms. Once we know the qualities of all the possible sets of norms, we intend to study the mechanisms that make a certain set of social norms become the dominant and used by the vast majority of the members of a society. Special attention will be paid on reputation mechanisms as a mean to control fraudulent behaviours.

Acknowledgments

This work was supported by the European Community under the FP6 programme (eRep project CIT5-028575 and OpenKnowledge project FP6-027253), by the project Autonomic Electronic Institutions (TIN2006-15662-C02-01), and partially supported by the Generalitat de Catalunya under the grant 2005-SGR-00093. Daniel Villatoro is supported by a CSIC predoctoral fellowship under JAE program. Jordi Sabater-Mir enjoys a Ramon y Cajal contract from the Spanish Government.

References

1. Bicchieri, C.: The Grammar of Society: The nature and Dynamics of Social Norms. Cambridge University Press, Cambridge (2006)
2. Saam, N.J., Harrer, A.: Simulating norms, social inequality, and functional change in artificial societies. Journal of Artificial Societies and Social Simulation 2(1) (1999)
3. Shoham, Y., Tenneholtz, M.: On the synthesis of useful social laws for artificial agent societies (preliminary report). In: Proceedings of the AAAI Conference, pp. 276–281 (1992)
4. Walker, A., Wooldridge, M.: Understanding the emergence of conventions in multi-agent systems. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi–Agent Systems, pp. 384–389. MIT Press, San Francisco (1995)
5. Grizard, A., Vercouter, L., Stratulat, T., Muller, G.: A peer-to-peer normative system to achieve social order. In: AAMAS 2006 Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN) (2006)
6. Axelrod, R.: An evolutionary approach to norms. The American Political Science Review 80(4), 1095–1111 (1986)
7. Sen, S., Airiau, S.: Emergence of norms through social learning. In: Proceedings of IJCAI 2007, pp. 1507–1512 (2007)
8. Gilbert, N.: Varieties of emergence. Edited transcript of the introductory talk given at the Workshop on Agent 2002 Social Agents: Ecology, Exchange, and Evolution Conference (October 2002)
9. Kittock, J.E.: The impact of locality and authority on emergent conventions: initial observations. In: AAAI 1994 Proceedings of the Twelfth National Conference on Artificial Intelligence. American Association for Artificial Intelligence, vol. 1, pp. 420–425 (1994)
10. Savarimuthu, B.T.R., Cranefield, S., Purvis, M., Purvis, M.: Role model based mechanism for norm emergence in artificial agent societies. In: Proceedings of the AAMAS 2007 Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN), Honolulu, Hawaii, USA (2007)

11. Excelente-Toledo, C.B., Jennings, N.R.: The dynamic selection of coordination mechanisms. Journal of Autonomous Agents and Multi-Agent Systems (2004)
12. Cecconi, F., Parisi, D.: Individual versus social survival strategies. Journal of Artificial Societies and Social Simulation 1(2) (1998)
13. Sen, S., Biswas, A., Debnath, S.: Believing others: Pros. and cons. (2000)
14. Villatoro, D., Sabater-Mir, J.: Norm selection through simulation in a resource-gathering society. In: Proceedings of 21st European Simulation and Modelling Conference (ESM) (2007)
15. Paolucci, M., Conte, R., Tosto, G.D.: A model of social organization and the evolution of food sharing in vampire bats. Adaptive Behavior 41(3), 223–239 (2006)
16. de Waal, F.: Good natured. Harvard University Press (1996)
17. Gusinde, M.: Los Indios de la Tierra del Fuego. CAEA (1982)
18. Rice, J.A.: Mathematical Statistics and Data Analysis. Duxbury Press (April 2001)

Transgression and Atonement

Kevin M. Knight, Deepthi Chandrasekaran, Aline Normoyle,
Ransom Weaver, and Barry G. Silverman

University of Pennsylvania, Philadelphia, PA 19104, USA
{kevinkn,dchandra,alinen,barryg}@seas.upenn.edu, ransom@ransomweaver.com

Abstract. This paper presents an approach to modeling social transgressions in agent based systems. The approach is intended to be abstract enough that it may be used with many different theories of transgression, apology, forgiveness, etc. In the first half of the paper, we consider what features of transgressions, people's emotional reactions to transgressions, and forgiveness are important, primarily by surveying social sciences literature. In the second half, we discuss an implementation of our approach in PMFserv, an agent based socio-cognitive modeling framework.

1 Introduction

In this paper we will present an approach to modeling social transgressions in agent based systems. By "social transgression" we mean an offense an agent can commit against social rules. Throughout this paper, the terms "transgression" and "offense" (and, similarly, "transgressor" and "offender") will be used interchangeably.

Our approach involves representing transgressions as abstract objects. However, the transgression objects themselves are not the main focus. Rather, they simply serve as a nexus for actions and relations between agents about the transgressions they represent.

The layout of this paper is as follows. In §2, we will discuss transgressions in general. In §3, we will discuss emotional reactions to transgressions. In §4, we will discuss issues of forgiveness. In §5, we will give an overview of PMFserv in preparation for the discussion of our implementation in §6.

2 Transgressions

In this section we will discuss issues with transgressions in general. One notable issue that we will not consider is perceptual mistakes. For example, we will not address situations in which an observer incorrectly blames an innocent party for a transgression or underestimates the effects. Such mistakes are beyond the scope of this paper.

All transgressions have a transgressor, a set of victims, and a set of effects. Effects are to be understood as the direct effects of the offending action, not the emotional effects on observers. Those are handled separately. For example, if Alan sets fire to Brad's house and burns it down, then the transgressor is Alan,

J.F. Hubner et al. (Eds.): COIN 2008, LNAI 5428, pp. 250–265, 2009.

the victims are Brad and whoever else has a stake in his house,[1] and the set of effects is that Brad's house has burned down and most everything inside has been damaged or destroyed.[2]

Now, one might argue that a transgression may have multiple transgressors. Take the example of a bank robbery executed by a gang with four members. Prima facie, this seems to be just such a transgression. However, we would consider it to be four separate transgressions – one for each robber. Or, in any case, we represent it as four separate transgression objects.

Metaphysical issues aside, we have pragmatic reasons for this method of dividing transgressions. First, it allows us to distinguish the different roles and levels of guilt of the different transgressors. For example, in a bank robbery, the gang member whose only job was to drive the getaway car may be held to a lower level of responsibility than the gang members who threatened the people in the bank with weapons.

Second, it allows us to keep relations between the different transgressors and the transgression separate. As we will discuss below, the transgression objects are used to keep track of such things as whether the transgressor committed the transgression intentionally. Since different transgressors may have had different levels of intent, even in what may be considered the same transgression, we use multiple objects to keep these relations separate.

Indeed, similar arguments can be made on the victim side. In many cases it is reasonable to create a separate transgression for each transgressor-victim pair. This allows for a very detailed level of accounting. However, in other cases such an approach may be infeasible. In the bank robbery example, there is a potentially very large number of victims, including the bank itself, all of its shareholders, everyone in the bank at the time of the robbery, and everyone whose money is lost. Whether to allow multiple victims is something that can be decided on a model by model or even transgression by transgression basis.

Beyond these basic properties of transgressions themselves, our transgression objects will keep track of some relations with the transgressor, relations with observers, properties of the effects, and relations between the transgressor and observers.

A transgression may be intentional or unintentional. That is, the transgressor may have performed the offensive action intentionally or unintentionally. For example, Alan may be angry at Brad and intentionally run his car into Brad's car. On the other hand, Alan may run into Brad's car accidentally.[3]

[1] Actually, the list of victims is much larger. For example, it may include owners of nearby houses, whose houses may also catch fire or may simply be damaged by soot. However, for the sake of simplicity, we will restrict the list of victims in our examples to only the most direct ones.

[2] Again, the list of effects is actually much larger, including such things as smoke damage to nearby structures, but we leave them out for the sake of simplicity. Furthermore, we do not count the emotional effects on Brad.

[3] Notice that intention is distinct from responsibility. That Alan ran his car into Brad's unintentionally does not imply that he is not responsible. For example, he might have been negligent in his driving.

The effects of a transgression may be active or inactive. Let us assume in our example of Alan crashing his car into Brad's that Brad's car is damaged and his arm is broken. Until his car is fixed (or replaced) and his arm heals, the effects of the transgression are active. Once those things happen, the effects are inactive.[4]

A related but separate issue is compensation. That is, some or all of the victims may have received compensation for the harm done to them. Compensation may or may not come from the transgressor and may or may not cause effects to become inactive. In our car crash example, compensation will probably come from Alan's insurance company rather than from Alan himself, and it will probably come in the form of money intended to cover repairs to or replacement of Brad's car as well as medical expenses. In this case, the compensation does not make the effects inactive. As noted before, the effects remain active until Brad's car is repaired or replaced and his arm heals.

A transgressor may or may not have apologized for the transgression. Apology is a complex subject, and there is much to say about it, both in terms of structure and effects. For the purposes of this paper, we will consider apology to be a black box. We will touch on the effects when we discuss forgiveness in §4.

A transgression may be forgivable or unforgivable. It seems that most people view most transgressions as, at least in principle, forgivable. Indeed, many transgressions are so minor that no one would consider them unforgivable. However, some people may view some transgressions as unforgivable, at least until some condition occurs (such as repentance of the transgressor).

Among forgivable transgressions, a transgression may be forgiven or unforgiven. This means that the observer in question may have forgiven the transgressor for the transgression. We will discuss forgiveness further in §4.

3 Emotional Reactions

Any observer could potentially have an emotional reaction to a transgression. This includes direct observers (i.e., those who directly perceive the transgression) and indirect observers (i.e., those who learn about the transgression by means other than direct perception, such as newspapers or other observers). It also includes those who have some relationship to those directly involved and those who have no such relationship.

For any transgression, we should expect that there is someone who would have a negative emotional reaction (e.g., anger or reproach) to observing it; otherwise, it would not be a transgression. However, this does not imply that everyone would have the same negative emotional reaction. For example, Wunderle points out that "Arabs believe it is imperative that negotiating partners respect each other's honor and dignity. To an American, losing face may be embarrassing, but to an Arab, it is devastating" [1, p. 36]. Even within a culture, there is considerable

[4] It is worth noting that not all effects can be made inactive. For example, in the case of a murder, the death may never be undone. However, the case of a transgression with permanent effects should not be confused with the case of an unforgivable transgression. Many transgressions with permanent effects may still be forgiven.

variation between individuals in the severity of their emotional reactions to the same transgression (see, e.g., [2,3,4]).

In addition to cultural and personality factors, the relationship between the observer and those directly involved in the transgression may affect the extent of the reaction. Gordijn et al. [5] and Yzerbyt et al. [6] studied emotional reactions of uninvolved observers to transgressions. Both found that negative emotional reactions to transgressions are significantly stronger when the victims are in the same group as the observer.

For many transgressions, we should expect that there is someone who would not view it as a transgression. For example, killing cattle is commonplace in America but taboo in India. Indeed, what is an egregious transgression to one may be a cause for celebration to another. Consider Bobby Fischer's reaction to the September 11, 2001 World Trade Center attack. During a radio interview in the Philippines hours after the event, he is reported as describing news of the attack as "wonderful" and saying that he "applaud[s] the act" [7].

In a nutshell, our framework must accommodate a wide variety of reactions to a transgression. In particular, it must handle different individuals viewing the same transgression as having different degrees of severity, as well as individuals who do not view the act as a transgression at all. However, since it is a framework for transgressions (and not acts in general), it need not handle the emotional reactions of those who do not view the act as a transgression (though it must not force a negative reaction upon them).

4 Forgiveness

We will consider what is sometimes called "offense-specific" forgiveness.[5] This is a relationship between three entities: a forgiver, a transgressor, and a transgression. The forgiver forgives the transgressor for the transgression. We are not concerned with whether or how the forgiver is connected to the transgressor or the transgression. However, the forgiver must be aware of the transgression and believe that the transgressor is in some way responsible for it.

We will divide forgiveness along two axes. The first axis is active versus passive forgiveness. *Active forgiveness* is where someone has made a conscious decision to forgive a transgressor; *passive forgiveness* is where no conscious decision has been made.[6]

The second axis is effective versus ineffective forgiveness. *Effective forgiveness* is where the negative emotions toward the transgressor resulting from the transgression have subsided; *ineffective forgiveness* is where the negative emotions have not subsided.

There are three possible combinations of these: effective active forgiveness, ineffective active forgiveness, and effective passive forgiveness. Ineffective passive

[5] Berry et al. [8] distinguish three types of forgiveness. *Offense-specific* is forgiveness of a specific person for a specific offense. *Dyadic* is forgiveness of a specific person for a history of offenses. *Dispositional* is forgiveness as an enduring personality trait.

[6] Active forgiveness should be understood as a private decision. Whether or not that decision is communicated to anyone else is a separate issue.

forgiveness is not really forgiveness, since neither the intent to forgive nor the desired result of forgiving is present.

Many (and perhaps most) definitions of forgiveness reflect a type of effective forgiveness.

- Subkoviak et al. define forgiveness as the "absence of negative affect, judgment, and behavior toward an offender and the presence of positive affect, judgment, and behavior toward the same offender" [9, p. 642].
- McCullough et al. define forgiveness as "the set of motivational changes whereby one becomes (a) decreasingly motivated to retaliate against an offending relationship partner, (b) decreasingly motivated to maintain estrangement from the offender, and (c) increasingly motivated by conciliation and goodwill for the offender, despite the offender's hurtful actions" [10, pp. 321-322].
- Berry et al. define forgiveness as "the juxtaposition or superimposition of strong, positive, other-oriented emotions over the negative emotions of unforgiveness" [8, p. 186].

Wohl et al. [11] caution against definitionally rejecting active ineffective forgiveness, describing it as *failed* forgiveness rather than non- or pseudo-forgiveness.[7] We will follow their lead on this point.

Three things are notable about the above definitions. First, none requires a conscious decision to forgive; thus, all are consistent with both active and passive forgiveness. Second, only the first definition requires a behavioral change (though we may expect behavioral changes to accompany the motivational or emotional changes required by the other two). Third, all involve both a decrease in negative emotions and an increase in positive emotions.

Regarding the third point, there is some evidence that the decrease in negative emotions is a separate process from the increase in positive emotions (see [3]). This is why an increase in positive emotions is not included in the definition of effective forgiveness.

Now let us consider the three cases of forgiveness, beginning with passive effective forgiveness. In this case, there has been no conscious decision to forgive but the negative emotions resulting from the transgression have subsided. One might expect that this state will occur with time, and there is evidence that this is correct.

McCullough et al. [3] showed that negative emotions (specifically avoidance and revenge motivation) associated with a transgression decrease linearly over time.[8] Moreover, while the rate of decrease varies from person to person, it does

[7] They say that for their subjects, "this profile of activities constitutes forgiveness even though forgiveness – as they conceive it – has failed to achieve the desired consequences (including those that researchers might stipulate)" [11, p. 558].

[8] Wohl and McGrath [4] confirmed these results and further noted that it is the *perceived* rather than actual amount of time that has passed that affects forgiveness. That is, avoidance and revenge motivation decrease with increases in perceived temporal distance. Since perceived temporal distance fluctuates, so do avoidance and revenge motivation.

not depend on the severity of the transgression. (On the other hand, the initial intensities of the negative emotions caused by the transgression do depend on its severity.) Importantly, McCullough et al. provide not only a theory of *whether* forgiveness will occur but also *when*.

Thus our framework must handle emotion decay, at least regarding the emotions caused by transgressions. However, while McCullough et al. have suggested that the rate of decay is linear and does not depend on the severity of the transgression, these assumptions are not built into the framework. We will discuss this in more detail in §5.3.

Next let us consider active effective forgiveness. This involves both a conscious decision to forgive and subsidence of negative emotions. Azar et al. [2] studied the effects of four factors on the propensity to forgive. The four factors were (1) whether the transgressor apologized, (2) whether the effects were still active, (3) whether the transgression was intentional, and (4) whether the transgressor was in the same social group as the potential forgiver. They found that the first two had major (and roughly equal) effects, the third a moderate effect, and the fourth an insignificant effect. Moreover, the effects combined additively.

Our transgression objects, as described in §2, make available all the information that Azar et al. designated as pertinent, including the social relationship between the observer and transgressor. Unfortunately, while Azar et al. provide insight into how these factors affect whether forgiveness will occur, they provide no insight into how the factors affect when it will occur. Nonetheless, our framework must be able to accommodate different theories about how these (or, indeed, other) factors affect both whether and when forgiveness will occur.

Finally, we will consider active ineffective forgiveness. This involves a conscious decision to forgive, but little or no subsidence of the negative emotions resulting from the transgression in question. This case should not be confused with the case in which the negative emotions caused by the transgression subside but are replaced by further transgressions.

Wohl et al. [11] refer to this case as failed forgiveness. In their study on different types of forgiveness, they identified a type in which the forgiver attempted to resume a positive relationship with the transgressor without ignoring or forgetting the transgression. In such cases, the relationships between forgiver and transgressor tended to deteriorate in the long run.

Unfortunately, we have very little insight into how or why such failed forgiveness might occur or what its precise effects are, including how and under what circumstances the relationship might deteriorate.

5 PMFserv

PMFserv (Performance Moderator Function Server) is a framework for modeling socio-cognitive agents. It includes a synthesis of about 100 best-of-breed models of personality, culture, values, emotions, stress, social relations, and group dynamics, as well as an integrated development environment for authoring and managing reusable archetypes and their task sets. For each agent, PMFserv

operates its perception, physiology, personality, and value system to determine stressors, grievances, tension buildup, the impact of speech acts, emotions, and various mobilization and collective and individual action decisions. PMFserv also manages social relationship parameters and thus macro-behavior (e.g., in collectives or crowds of agents) emerges from individuals interactions and micro-decisions.

PMFserv is in use by an intelligence agency to model diplomatic decisions of world leaders for which it has passed statistical correspondence tests showing it is significantly in agreement with their decision making [12,13]. PMFserv has also reached the level where it can realistically simulate ethno-political conflicts among regional leaders and their followers vying over control of contested resources and assets. For more detailed accounts of PMFserv, including validation studies for application in the Far East, Middle East, Africa, and North America, see [12,13,14,15].

5.1 Goals, Standards, and Preferences

Agents' cultural values and personality traits are modeled in PMFserv by goal, standard, and preference (GSP) trees. These are multi-attribute value structures where each tree node is weighted with Bayesian importance weights.

Preferences are long term desires for world situations and relations. In the implementation we describe below, relevant preferences include whether the agent has a materialistic, symbolistic, or humanistic vision of the future.

Goals cover short-term needs and motivations that implement progress toward preferences. Goals relevant to the implementation we describe below include needs for belonging, esteem, and safety.

Standards define the methods an agent is willing to use to satisfy its goals and preferences. These include concerns with conformance (to society), relationship versus task focus, sensitivity to life, willingness to use violence, concern with honesty, respect for authority, and narrow self-interest versus concern for the greater good.

The example goals, standards, and preferences just mentioned are summarized in Table 1. It should be noted that these are just examples which will be relevant in §6 and by no means exhaust the set of possible goals, standards, and preferences.

In addition to Bayesian importance weights, the nodes in the GSP trees have positive and negative activations (represented by values in $[0, 1]$, where

Table 1. Example goals, standards, and preferences

Goals	Standards	Preferences
Belonging	Conformance	Humanistic
Esteem	Relationship vs. task focus	Materialistic
Safety	Sensitivity to life	Symbolistic
	Use of violence	
	Honesty	
	Respect for authority	
	Self-interest vs. greater good	

0 indicates no activation and 1 full activation). A node becomes activated when an agent takes an action related to that node. For example, if an agent takes an action involving deceit, then the node representing its standard of honesty would be negatively activated. These activations are used to calculate the agent's current emotional state. GSP trees and how they relate to emotions in PMFserv have been discussed at length elsewhere (e.g., [12,14]), and we will not reproduce that discussion here.

5.2 Objects

In addition to managing agents, PMFserv manages objects (representing both agents and non-agents, such as cars or locations), including when and how they may be perceived and acted on by agents. PMFserv implements affordance theory [16], meaning that each object applies perception rules to determine how it should be perceived by each perceiving agent.[9] Objects then reveal the actions (and the potential results of performing those actions) afforded to the agent. For example, an object representing a car might afford a driving action which can result in moving from one location to another.

Notably, objects need not be concrete. PMFserv makes no metaphysical assumptions about its objects. Abstract objects, such as plans and obligations, may be represented just as easily as concrete objects.

Objects have a state, which is a set of properties. For example, an object representing a car might have a make, model, color, sale price, etc. Additionally, objects have a set of perceptual types. Each perceptual type has a perceptual rule associated with it which is used to determine whether that type is perceived. For example, a car might have a *buyable* perceptual type which indicates whether an agent perceives the car as something it can purchase. The perceptual rule associated with the type might compare the sale price of the car with the amount of money the agent has (as well as considering whether the car is owned by someone else and is for sale). Assuming that the agent perceives the car as *buyable*, the action *buy* would be afforded with the result that the car changes ownership, the current owner's money increases, the agent's money decreases, and the agent's emotional state changes appropriately.

In addition to binary perceptual types, there are "continuous" perceptual types. Rather than an agent viewing an object as either having this sort of perceptual type or not, agents view an object as having it to a certain degree between 0 and 1. The degree of perception and the precise meaning of the degree are determined by perceptual rules. For example, one agent might view another agent as more or less of a friend.

Furthermore, groups of perceptual types for an object may be designated as mutually exclusive. That is, at most one of such a group may be perceived at a time by an agent. A perceptual type may be in at most one such group for an object.

[9] This approach was chosen for pragmatic rather than philosophical reasons. See [14] for a discussion of the reasons.

5.3 Emotion Decay

In PMFserv, whenever an event occurs which should elicit an emotional reaction from an agent, the agent notes the event along with its initial emotional impact and assigns a decay function to it. An agent's emotional state at any given time is determined by the initial impact, decay function, and age of each event that it has stored.

Each decay function takes the initial impact and age of an event and returns the decayed impact, i.e., the impact the event will have after a certain amount of time has passed. In principle, there is no limitation to the nature of the function, though under normal circumstances it should be monotonically decreasing.

Each agent has its own decay policy which assigns decay functions to events. Like the decay functions, there are no real limits to their nature. A decay policy could, for example, assign the same decay function to all events; or it could assign decay functions based on properties of the events.

6 Implementation

In PMFserv, transgressions are represented as abstract objects. They are dynamically created when an agent transgresses. This requires the scenario designer (ideally in consultation with a subject matter expert) to decide which actions count as transgressions and what impact they will have on observers. These may vary significantly between scenarios since they depend on the actions that are available and the sorts of agents being modeled.

In our current implementation, we are modeling Arab villagers and US soldiers in an Iraqi village. Emphasis in this article is on transgressions that US soldiers can commit against villagers and how they may atone (though the implementation also handles transgressions between villagers). While there are in fact a vast number of such transgressions, for this discussion we will concentrate on three examples: rude and untactful speech (*adeb*), searching a home without dogs, and searching a home with dogs. Both cases of searching refer to soldiers searching a villager's home by force or the threat of force.

Before discussing the transgressions any further, we will discuss some simplifying assumptions we are making, mostly with respect to perception and communication.

The first assumption is that all transgressions are perceived immediately by everyone. This does not necessarily mean that everyone directly perceives every transgression, simply that everyone is immediately aware. Essentially, we are assuming that communication about transgressions within the village is complete and effectively instantaneous.

The second assumption is that transgression objects have only one victim. Thus transgressions which have multiple victims will be represented by multiple transgression objects each with one victim.

The third assumption is that only Arab villagers are offended. That is, we are not representing transgressions that villagers can commit against soldiers or soldiers against each other.

Now let us put this into a more formal representation. Let T be the set of transgressions, A be the set of agents, E be the set of effects, and $\langle TP, \prec, d\rangle$ be a structure representing time, where TP is the set of timepoints, $\prec$ is a linear ordering, and d is a distance function. We will represent a transgression $\tau \in T$ as a quadruple $\langle o, v, e, t\rangle$, where $o \in A$ is the offender (or transgressor), $v \in A$ is the victim, $e \subseteq E$ is the set of effects, and $t \in TP$ is the time at which the transgression occurred. For a transgression τ, we will denote these as τ_o, τ_v, τ_e, and τ_t, respectively.

Based on the discussion in §2 we will define the following predicates. For $\tau \in T$, $\alpha \in A$, and $t_0, t_1 \in TP$,

- $intentional(\tau)$ is true iff τ was intentional,
- $apologized(\tau, t_1)$ is true iff τ_o apologized for τ at some time $t_0 \preceq t_1$,
- $active(\tau, t_1)$ is true iff τ_e are still active at t_1, and
- $forgivable(\tau, \alpha)$ is true iff observer α views τ as forgivable.

We will define some more functions and predicates after further discussion.

Now that we have stated what our transgressions are, we must say what their impact on observers will be. That is, we must associate each transgression with a set of GSP activations that will be afforded to observers. We can represent afforded activations as a vector in $[0, 1]^{2n}$, where n is the number of GSP nodes. (The vector is of length $2n$ because it must contain both positive and negative activations for each node.)

To facilitate combination of such vectors, we define the bounded addition operator, written $\oplus$. For $x, y \in \mathbb{R}$, we define scalar bounded addition as follows.

$$x \oplus y = \max(0, \min(1, x + y)) \tag{1}$$

We define vector bounded addition as element-wise scalar bounded addition.

As a convenient way to organize these in our implementation, each transgression is assigned an intensity in each of the following categories: *faux pas*, *taboo*, *violent*, *materialistic*, and *deceitful*.[10] Intensities range from zero to one, and it is common for transgressions to have non-zero intensities in multiple categories. For example, a mugging is both violent and materialistic. The meaning of each category is summarized in Table 2.

Table 2. Transgression categories

Faux pas	**Taboo**	**Violent**	**Materialistic**	**Deceitful**
Relationship focus	Relationship focus	Sensitivity to life	Self-interest	Honesty
Conformance	Conformance	Use of violence	Respect for authority	
Belonging	Belonging	Belonging	Materialistic	
Esteem	Esteem	Safety		
	Symbolistic			

[10] These categories should not be taken as an authoritative taxonomy of transgressions. They were chosen because they correspond well to nodes in the GSP trees used in the current scenario. For research on moral categories, see, e.g., [17].

Faux pas are comparatively minor transgressions related to etiquette. Examples include rude speech and inappropriate dress. These afford activations to an observer's GSP nodes related to focusing on relationships, conformance, and concerns with belonging and esteem. Let $fp \in [0,1]^{2n}$ be the activations afforded by a faux pas transgression.

Taboo transgressions are similar in nature to faux pas, though they are generally more serious. Examples include marrying a sibling and making blasphemous statements. These afford activations to the same GSP nodes as faux pas plus those related to symbolistic concerns. Let $tb \in [0,1]^{2n}$ be the activations afforded by a taboo transgression.[11]

Violent transgressions can range from the relatively minor to the extremely serious. Both actual violence and the threat of violence are included. Examples include slapping someone in the face and setting off a bomb in a crowded marketplace. These afford activations to an observer's GSP nodes related to sensitivity to human life, the use of violence, and concerns with belonging and safety. Let $vi \in [0,1]^{2n}$ be the activations afforded by a violent transgression.

Materialistic transgressions are those having to do with property. This includes damaging, destroying, and stealing property. Examples include vandalism and theft. These afford activations to an observer's GSP nodes related to self-interest, respect for authority, and materialistic concerns. Let $ma \in [0,1]^{2n}$ be the activations afforded by a materialistic transgression.

Deceitful transgressions are those relating to honesty. They include everything from little white lies to major fraud. These afford activations to an observer's GSP nodes related to honesty. Let $de \in [0,1]^{2n}$ be the activations afforded by a deceitful transgression.[12]

Intensities for our example transgressions can be found in Table 3.[13] *Adeb* is a relatively minor faux pas. Searching a home (with or without dogs) involves the threat of violence and offense against property. Searching a home with dogs also involves elements of taboo, since dogs are considered unclean by many Arabs.

To denote the intensity of a transgression in each category, we will define five functions from T to $[0,1]$: $fauxpas$, $taboo$, $violent$, $materialistic$, and $deceitful$. The base impact a transgression $\tau \in T$ will have on an observer is defined by the following equation.

$$I_b(\tau) = fauxpas(\tau) \cdot fp \oplus taboo(\tau) \cdot tb \oplus violent(\tau) \cdot vi \oplus \\ \oplus materialistic(\tau) \cdot ma \oplus deceitful(\tau) \cdot de \quad (2)$$

[11] Faux pas and taboo are notably similar categories. The main difference is that taboo transgressions violate deeply held convictions. While a faux pas transgression might result in feelings of annoyance or perhaps even mild contempt, a taboo transgression would more likely result in feelings of anger or even disgust. Consider the difference between addressing someone in an inappropriate way and throwing feces at that person.

[12] This category is currently a placeholder. At this time, agents are not able to take deceptive actions in PMFserv.

[13] The intensities were assigned by our fourth author, an Arabist, who reviewed the transgressions.

Table 3. Example transgression intensities

	Faux pas	Taboo	Violent	Materialistic	Deceitful
Adeb	0.1	0.0	0.0	0.0	0.0
Searching a home	0.1	0.0	0.2	0.1	0.0
Searching a home with dogs	0.1	0.5	0.5	0.1	0.0

The initial impact is affected by two other factors: the relationship of the observer to the victim and whether the transgression was intentional.

As noted in §3, the relationship of the observer to the victim can affect the impact of a transgression. In particular, the closer the relationship, the more severe the impact. We consider four types of relationships: whether the observer is the victim, the victim's kin, in the same group as the victim, or in a group with at least neutral relations with the victim's group. If the observer does not share one of these relationships with the victim, then there will be no relationship based impact. For $\tau \in T$ and $\alpha \in A$, we will denote the impact of α's relationship to τ_v by $I_r(\tau, \alpha) \in [0, 1]^{2n}$.

Based on studies by Azar et al. [2] (see the discussion in §4), we give additional initial impact if the transgression was intentional. For $\tau \in T$, we denote this impact by $I_n(\tau) \in [0, 1]^{2n}$, where $I_n(\tau) = \langle 0, \ldots, 0 \rangle$ if $intentional(\tau)$ is not true.

Thus for $\tau \in T$ and $\alpha \in A$, the initial impact I_i of α observing τ is described by the following equation.[14]

$$I_i(\tau, \alpha) = I_b(\tau) \oplus I_r(\tau, \alpha) \oplus I_n(\tau) \tag{3}$$

Of course, the actual emotional effect τ will have on α is a function of $I_i(\tau, \alpha)$, α's personality, and α's prior emotional state. For example, *adeb* will not bother someone who is not concerned with relationships, conformance, belonging, or esteem. On the other hand, someone who is concerned with one or more of those will likely be bothered, though probably not too much since *adeb* is a minor transgression at worst.

All of these factors are implemented as perceptual types on the transgression object. The categories are implemented as continuous perceptual types (where the perception levels are the intensities from Table 3), the relationship is implemented as a mutually exclusive group of binary perceptual types, and the intentionality is represented as a single binary perceptual type. These perceptual types afford *perceive* actions, which are performed automatically when the object is introduced.

There are two binary perceptual types on the transgression object which afford substantive actions. The first is perceivable if the effects of the transgression are still active and affords the action *remove effects*. The second is perceivable only to the transgressor if he has not apologized and affords the action *apologize*.

[14] We aggregate by addition for the sake of simplicity. It is not clear from the literature, for example, whether the effect of the relationship is additive or multiplicative – we know only that it increases the impact.

The two actions are similar in effect. Both reduce the emotional impact of the transgression on observers, thus decreasing the time it takes to forgive. This is based on the claim of Azar et al. [2] (see the discussion in §4) that whether the effects of the transgression are still active and whether the transgressor has apologized significantly contribute to the likelihood of forgiveness. Notably, neither action can be performed multiple times for the same transgression.

The *apologize* action has an added dimension in our model since some transgressions require atonement more complicated than a simple verbal apology. Consider, for example, the ritual of "blood money" paid for an offense resulting in death in Arab cultures.[15] To this end, we include atonement objects, which encapsulate the steps necessary for atonement. Once all the steps have been completed, the effect is that of having apologized.

Formally, performing the *remove effects* or *apologize* on transgression $\tau \in T$ at time $t \in TP$ has the effect of making $active(\tau, t^{'})$ false or $apologized(\tau, t^{'})$ true for all $t^{'} \in TP$ such that $t \preceq t^{'}$. We implement the reduction in emotional impact by associating coefficients with each as follows.

$$C_e(\tau, t) = \begin{cases} 0 & \text{if } active(\tau, t) \\ -\frac{1}{2} & \text{otherwise} \end{cases} \tag{4}$$

$$C_a(\tau, t) = \begin{cases} -\frac{1}{3} & \text{if } apologized(\tau, t) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

These are used to adjust the impact as follows.

$$I(\tau, \alpha, t) = (1 + C_e(\tau, t) + C_a(\tau, t)) \cdot I_i(\tau, \alpha) \tag{6}$$

In other words, removing the effects reduces the impact by half and apologizing reduces it by one third.[16]

Furthermore, there is the question of how the impact decays over time. Each agent is assigned a "grudge factor" ranging from 0 to 1 and indicating for how long the agent will hold a grudge. The higher the grudge factor, the longer it will take the agent to forgive a transgression. In practical terms, this determines the emotion decay function for that agent. Following McCullough et al. [3], all decay functions are linear, and the grudge factor simply serves to determine

[15] The ritual may be fairly elaborate as, for example, described by Irani and Funk [18]. In this case, the family of the offender must seek the help of a delegation of local leaders, esteemed mediators, and other notables, who will hear the grievances of the victim's family and determine what constitutes an appropriate payment of "blood money" in the case at hand. Then the offending and offended families gather together for a ritual shaking of hands. Then the family of the victim serves bitter coffee to the family of the offender to demonstrate forgiveness. Finally, the offending family serves a meal to the offended family.

[16] The coefficients are guesses on our part. Their additivity is supported by Azar et al. [2]; however, we could find no support in the literature for particular values.

the slope (with a lower grudge factor indicating a steeper slope).[17] Slopes range from -1 (indicating more or less instantaneous forgiveness) to 0 (indicating no forgiveness).

The only exception is for unforgivable transgressions. For those transgressions, emotions do not decay. Such transgressions are rare, and none of our examples fall into this category. Whether a transgression is unforgivable is implemented as a binary perceptual type on the transgression object.

For $\alpha \in A$, let us denote the slope of α's decay function by $\alpha_d \in [-1, 0]$. Thus for $\tau \in T$ and $t \in TP$ such that $\tau_t \preceq t$, the amount that the impact of τ should have decayed by t is described by the following equation.

$$\delta(\tau, \alpha, t) = \begin{cases} \alpha_d \cdot d(t, \tau_t) & \text{if } forgivable(\tau, \alpha) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

(where d is a temporal distance function). Now we define the decayed impact of τ on α at t as

$$D(\tau, \alpha, t) = I(\tau, \alpha, t) \oplus \boldsymbol{\delta}(\tau, \alpha, t) \tag{8}$$

where $\boldsymbol{\delta}(\tau, \alpha, t)$ is a vector in $\mathbb{R}^{2n}$ whose elements are all $\delta(\tau, \alpha, t)$.

Once the emotional impact of a transgression has decayed to nothing, then the agent has effectively forgiven the transgression. In other words, for $\tau \in T$, $\alpha \in A$, and $t \in TP$ such that $\tau_t \preceq t$, α has effectively forgiven τ_o for τ at t if $D(\tau, \alpha, t) = \langle 0, \ldots, 0 \rangle$.

Consider a few examples with the sample transgressions mentioned earlier. If a soldier commits *adeb*, most villagers will have a slightly negative emotional reaction. However, for the most part they will get over it quickly, especially if the soldier apologizes.

Searching a home is a more serious transgression, involving violent and materialistic elements as well as breaching etiquette. Villagers will have a much stronger reaction than to *adeb*. However, once the effects become inactive and the soldier apologizes, most villagers should forgive the soldier for that particular transgression within a few weeks (though the villagers may still have negative emotions about the soldier if he has committed further transgressions).

Searching a home with a dog is a considerably more severe transgression than either of the previous two. In addition to the effects of simply searching a home, this violates the taboo of bringing a dog into a home. Thus the emotional impact on the villagers will be considerably stronger. Even once the effects have been removed and the soldier has apologized, forgiveness may take quite some time, perhaps several months (with the same qualification as before). And without an apology forgiveness will take considerably longer.

[17] Our assignment of grudge factors, and thus decay rates, to agents is somewhat arbitrary. We made what we consider to be plausible guesses, but as far as we can tell, the literature is largely silent on this issue.

7 Conclusion

We have presented an approach to modeling transgressions in agent based systems. To this end we have discussed a number of considerations relevant to any model of emotional reaction to and forgiveness of transgressions. And we have described an implementation in PMFserv.

There are still many open issues on this topic. We did not consider the question of observers having incomplete or incorrect information about a transgression. Similarly, there are issues we did not consider with communication, such as agents (intentionally or unintentionally) introducing their own biases when informing others of a transgression. In the real world these are very common cases.

The issue of collective responsibility remains open. That is, how observers attribute blame to groups for individual transgressions. For example, when a US soldier commits a transgression, how much will observers blame the soldier himself versus the US military versus the US as a whole?

Another interesting issue we did not consider is apology. There is a great deal to say on the subject, particularly regarding the effectiveness of different apology strategies and the likelihood of an apology being rejected.

Finally, beyond conceptual issues, for any approach to modeling transgressions to be really useful, actual transgressions and their impacts must be cataloged.

References

1. Wunderle, W.: How to negotiate in the Middle East. Military Review 87(2), 33–37 (2007)
2. Azar, F., Mullet, E., Vinsonneau, G.: The propensity to forgive: Findings from Lebanon. Journal of Peace Research 36(2), 169–181 (1999)
3. McCullough, M.E., Fincham, F.D., Tsang, J.-A.: Forgiveness, forbearance, and time: The temporal unfolding of transgression-related interpersonal motivation. Journal of Personality and Social Psychology 84(3), 540–557 (2003)
4. Wohl, M.J.A., McGrath, A.L.: The perception of time heals all wounds: Temporal distance affects willingness to forgive following an interpersonal transgression. Personality and Social Psychology Bulletin 33(7), 1023–1035 (2007)
5. Gordijn, E.H., Wigboldus, D., Yzerbyt, V.: Emotional consequences of categorizing victims of negative outgroup behavior as ingroup or outgroup. Group Processes & Intergroup Relations 4(4), 317–326 (2001)
6. Yzerbyt, V., Dumont, M., Wigboldus, D., Gordijn, E.: I feel for us: The impact of categorization and identification on emotions and action tendencies. British Journal of Social Psychology 42(4), 533–549 (2003)
7. Bamber, D., Hastings, C.: Bobby Fischer speaks out to applaud Trade Center attacks. Sunday Telegraph (London), 17 (December 2, 2001)
8. Berry, J.W., Worthington, E.L., O'Connor, L.E., Parrott, L., Wade, N.G.: Forgivingness, vengeful rumination, and affective traits. Journal of Personality 73(1), 183–225 (2005)
9. Subkoviak, M.J., Enright, R.D., Wu, C.R., Gassin, E.A., Freedman, S., Olson, L.M., Sarinopoulos, I.: Measuring interpersonal forgiveness in late adolescence and middle adulthood. Journal of Adolescence 18(6), 641–655 (1995)

10. McCullough, M.E., Worthington, E.L., Rachal, K.C.: Interpersonal forgiving in close relationships. Journal of Personality and Social Psychology 73(2), 321–336 (1997)
11. Wohl, M.J.A., Kuiken, D., Noels, K.A.: Three ways to forgive: A numerically aided phenomenological study. British Journal of Social Psychology 45(3), 547–561 (2006)
12. Silverman, B.G., Bharathy, G.K., Nye, B., Eidelson, R.J.: Modeling factions for "effects based operations": Part I – leaders and followers. Computational & Mathematical Organization Theory 13(4), 379–406 (2007)
13. Silverman, B.G., Bharathy, G.K., Nye, B., Smith, T.: Modeling factions for "effects based operations": Part II – behavioral game theory. Computational & Mathematical Organization Theory 14(2), 120–155 (2008)
14. Silverman, B.G., Johns, M., Cornwell, J.B., O'Brien, K.: Human behavior models for agents in simulators and games: Part I: Enabling science with PMFserv. Presence: Teleoperators and Virtual Environments 15(2), 139–162 (2006)
15. Silverman, B.G., Bharathy, G.K., O'Brien, K., Cornwell, J.B.: Human behavior models for agents in simulators and games: Part II: Gamebot engineering with PMFserv. Presence: Teleoperators and Virtual Environments 15(2), 163–185 (2006)
16. Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton Mifflin, Boston (1979)
17. Haidt, J.: The new synthesis in moral psychology. Science 316(5827), 998–1002 (2007)
18. Irani, G.E., Funk, N.C.: Rituals of reconciliation: Arab-Islamic perspectives. Arab Studies Quarterly 20(4), 53–74 (1998)

Author Index