11th International Workshop on

Coordination, Organization, Institutions and Norms in Agent Systems

Lyon, France,
30th August - 2nd September 2010

Workshop Notes
Preface

The development of complex distributed AI systems with heterogeneous and diverse knowledge is a challenge. System components must interact, coordinate and collaborate to manage scale and complexity of task environments targeting persistency and maybe, evolution of systems. Managing scale and complexity requires organized intelligence; in particular intelligence manifested in organizations of agents, by individual strategies or collective behaviour. System architects have to consider: the inter-operation of heterogeneously designed, developed or discovered components (agents, objects/artefacts, services provided in an open environment); inter-connection which cross legal, temporal, or organizational boundaries; the absence of global objects or centralised controllers; the possibility that components will not comply with the given specifications; and embedding in an environment which is likely to change, with possible impact on individual and collective objectives.

The convergence of the requirement for intelligence with these operational constraints demands: coordination, the collective ability of heterogeneous and autonomous components to arrange or synchronise the performance of specified actions in sequential or temporal order; rational and open organization, a formal structure supporting or producing intentional forms of coordination, capable of managing changes in the environment in which it operates; institution, an organization where the performance of designated actions by empowered agents produces conventional outcomes; and norms, standards or patterns of behaviour in an institution established by decree, agreement, emergence, and so on.

The automation and distribution of intelligence is the subject of study in autonomous agents and multi-agent systems; the automation and distribution of intelligence for coordination, organization, institutions and norms in Agent Systems (COIN), in its eleventh edition. The COIN@MALLOW 2010 workshop is part of the COIN series of workshops http://www.pcs.usp.br/coin/.

This edition of COIN received fourteen high quality submissions, describing works by researchers coming from nine different countries, eight of which have been selected by the Programme Committee as regular papers and two of which have been selected by the Programme Committee as position papers. Each paper received at least three reviews in order to supply the authors with helpful feedback that could stimulate the research as well as foster discussion. COIN@AAMAS2010 and COIN@MALLOW2010 post-proceedings will be published soon in a single Springer LNCS volume.

We would like to thank all authors for their contributions, the members of the Steering Committee for the valuable suggestions and support, and the members of the Programme Committee for their excellent work during the reviewing phase.

August 4th, 2010

Nicoletta Fornara, George Vouros
Workshop Organisers

Nicoletta Fornara  University of Lugano, Switzerland
George Vouros  University of the Aegean, Greece

Programme Committee

Alexander Artikis  National Centre for Scientific Research Demokritos, Greece
Guido Boella  University of Torino, Italy
Olivier Boissier  ENS Mines Saint-Etienne, France
Rafael Bordini  Federal University of Rio Grande do Sul, Brazil
Amit Chopra  University of Trento, Italy
Antonio Carlos da Rocha Costa  Univ. Federal do Rio Grande FURG, Brazil
Marina De Vos  University of Bath, UK
Virginia Dignum  Delft University of Technology, The Netherlands
Jomi Fred Hubner  Federal University of Santa Catarina, Brazil
Christian Lemaitre  Universidad Autonoma Metropolitana, Mexico
Henrique Lopes Cardoso  Universidade do Porto, Portugal
Eric Matson  Purdue, USA
John-Jules Meyer  Utrecht University, The Netherlands
Pablo Noriega  IIIA-CSIC, Spain
Eugenio Oliveira  Universidade do Porto, Portugal
Andrea Omicini  University of Bologna, Italy
Sascha Ossowski  URJC, Spain
Julian Padget  University of Bath, UK
Jeremy Pitt  Imperial College, London, UK
Juan Antonio Rodriguez Aguilar  IIIA-CSIC, Spain
Jaime Sichman  University of Sao Paulo, Brazil
Munindar P. Singh  North Carolina State University, USA
Viviane Torres da Silva  Universidade Federal Fluminense, Brazil
Kostas Stathis  Royal Holloway, University of London, UK
Paolo Torroni  University of Bologna, Italy
Leon van der Torre  University of Luxembourg, Luxembourg
Birna van Riemsdijk  Delf University of Technology, The Netherlands
Wamberto Vasconcelos  University of Aberdeen, UK
Javier Vazquez-Salceda  University Politecnica de Catalunya, Spain
Mario Verdicchio  University of Bergamo, Italy
Danny Weyns  Katholieke Universiteit Leuven, Germany
Pinar Yolum  Bogazici University, Turkey
**Additional Reviewers**

Luciano Coutinho  
Universidade de Campinas, Brazil

Akin Gunay  
Bogazici University, Turkey

Ozgur Kafali  
Bogazici University, Turkey

**Steering Committee**

Guido Boella  
University of Torino, Italy

Olivier Boissier  
ENS Mines Saint-Etienne, France

Nicoletta Fornara  
University of Lugano, Switzerland

Christian Lemaitre  
Universidad Autonoma Metropolitana, Mexico

Eric Matson  
Purdue University, USA

Pablo Noriega  
Artificial Intelligence Research Institute, Spain

Sascha Ossowski  
Universidad Rey Juan Carlos, Spain

Julian Padget  
University of Bath, UK

Jeremy Pitt  
Imperial College London, UK

Jaime Sichman  
University of Sao Paulo, Brazil

Wamberto Vasconcelos  
University of Aberdeen, UK

Javier Vzquez Salceda  
Universitat Politecnica de Catalunya, Spain

George Vouros  
University of the Aegean, Greece
# Table of Contents

Normative Monitoring: Semantics and Implementation .......................... 1  
*Sergio Alvarez-Napayoa, Huib Aldewereld, Javier Vazquez, Frank Dignum*

Controlling multi-party interaction within normative multi-agent organizations .......................... 17  
*Olivier Boissier, Flavien Balbo, Fabien Badeig*

Norm Refinement and Design through Inductive Learning ....................... 33  
*Domenico Corapi, Marina De Vos, Julian Padget, Alessandra Russo, Ken Satoh*

Norm enforceability in Electronic Institutions? .................................. 49  
*Natalia Criado, Estefania Argente, Antonio Garrido, Juan A. Gimeno, Francesc Igual, Vicente Botti, Pablo Noriega, Adriana Giret*

Towards a Normative BDI Architecture for Norm Compliance ............... 65  
*Natalia Criado, Estefania Argente, Pablo Noriega, Vicent Botti*

Generating Executable MAS-Prototypes from SONAR Specifications ....... 82  
*Michael Köhler-Bußmeier, Matthias Wester-Ebbinghaus, Daniel Moldt*

Embodied Organizations: a unifying perspective in programming Agents, Organizations and Environments ........................................... 98  
*Michele Piunti, Olivier Boissier, Jomi F. Häubner, Alessandro Ricci*

Group intention = social choice + commitment ................................. 115  
*Marija Slavkovik, Guido Boella, Gabriella Pigozzi, Leon van der Torre*

Position Papers

*Matteo Baldoni, Cristina Baroglio, Federico Bergenti, Antonio Bossalatte, Elisa Marengo, Maurizio Martelli, Viviana Mascardi, Luca Padovani, Viviana Patti, Alessandro Ricci, Gianfranco Rossi, Andrea Santi*

Contextual Integrity and Privacy Enforcing Norms for Virtual Communities ........................................ 150  
*Yann Krupa, Laurent Vercouter*
Normative Monitoring:
Semantics and Implementation

Sergio Alvarez-Napagao¹, Huib Aldewereld²,
Javier Vázquez-Salceda¹, and Frank Dignum²

¹ Universitat Politècnica de Catalunya
{salvarez,jvazquez}@lsi.upc.edu
² Universiteit Utrecht
{huib,dignum}@cs.uu.nl

Abstract. The concept of Normative Systems can be used in the scope of Multi-Agent Systems to provide reliable contexts of interactions between agents where acceptable behaviour is specified in terms of norms. Literature on the topic is growing rapidly, and there is a considerable amount of theoretical frameworks for normative environments, some in the form of Electronic Institutions. Most of these approaches focus on regulative norms rather than on substantive norms, and lack a proper implementation of the ontological connection between brute events and institutional facts. In this paper we present a formalism for the monitoring of both regulative (deontic) and substantive (constitutive) norms based on Structural Operational Semantics, its reduction to Production Systems semantics and our current implementation compliant to these semantics.

1 Introduction

In recent years, several researchers have argued that the design of multi-agent systems (MAS) in complex, open environments can benefit from social abstractions in order to cope with problems in coordination, cooperation and trust among agents, problems which are also present in human societies. One of such abstractions is Normative Systems. Research in Normative Systems focuses on the concepts of norms and normative environment (which some authors refer to as institutions) in order to provide normative frameworks to restrict or guide the behaviour of (software) agents. The main idea is that the interactions among a group of (software) agents are ruled by a set of explicit norms expressed in a computational language representation that agents can interpret. Although some authors only see norms as inflexible restrictions to agent behaviour, others see norms not as a negative, constraining factor but as an aid that guides the agents’ choices and reduces the complexity of the environment, making the behaviour of other agents more predictable.

Until recently, most of the work on normative environments works with norm specifications that are static and stable, and which will not change over time. Although this may be good enough from the social (institutional) perspective, it is not appropriate from the agent perspective. During their lifetime, agents may enter and leave several interaction contexts, each with its own normative framework. Furthermore they may be operating in contexts where more than one normative specification applies. So we need
mechanisms where normative specifications can be added to the agents’ knowledge base at run-time and be practically used in their reasoning, both to be able to interpret institutional facts from brute ones (by using constitutive norms to, e.g. decide if killing a person counts as murder in the current context) and to decide what ought to be done (by using regulative norms to, e.g. prosecute the murderer). In this paper we propose to use production systems to build a norm monitoring mechanism that can be used both by agents to perceive the current normative state of their environment, and for these environments to detect norm violations and enforce sanctions. Our basic idea is that an agent can configure, at a practical level, the production system at run-time by adding abstract organisational specifications and sets of counts-as rules.

In our approach, the detection of normative states is a passive procedure consisting in monitoring past events and checking them against a set of active norms. This type of reasoning is already covered by the declarative aspect of production systems, so no additional implementation in an imperative language is needed. Using a forward-chaining rule engine, events will automatically trigger the normative state - based on the operational semantics - without requiring a design on how to do it.

Having 1) a direct syntactic translation from norms to rules and 2) a logic implemented in an engine consistent with the process we want to accomplish, allows us to decouple normative state monitoring from the agent reasoning. The initial set of rules we have defined is the same for each type of agent and each type of organisation, and the agent will be able to transparently query the current normative state at any moment and reason upon it. Also this decoupling helps building third party/facilitator agents capable of observing, monitoring and reporting normative state change or even enforcing behaviour in the organisation.

In this paper we present a formalism for the monitoring of both regulative (deontic) and substantive (constitutive) norms based on Structural Operational Semantics (Section 2), its reduction to Production Systems semantics (Section 3) and our current implementation compliant to these semantics (Section 4). In Section 5 we compare with other related work and provide some conclusions.

2 Formal Semantics

In this section we discuss the formal semantics of our framework. First, in section 2.1, we give the semantics of institutions as the environment specifying the regulative and constitutive norms. Then, in section 2.2, we describe the details of how this institution evolves over time based on events, and how this impacts the monitoring process. This formalisation will be used in section 3 as basis of our implementation.

Through this paper, we will use as an example the following simplified traffic scenario:

1. A person driving on a street is not allowed to break a traffic convention.
2. In case (1) is violated, the driver must pay a fine.
3. In a city, to exceed 50kmh counts as breaking a traffic convention.
2.1 Preliminary definitions

Before giving a formal definition of institutions (see definition 4), we first define the semantics of the regulative and constitutive norms part of that institution (in definitions 1 and 3, respectively).

We assume the use of a predicate based propositional logic $L_O$ with predicates and constants taken from an ontology $O$, and the logical connectives $\{\neg, \lor, \land\}$. The set of all possible well-formed formulas of $L_O$ is denoted as $wff(L_O)$ and we assume that each formula from $wff(L_O)$ is normalised in Disjunctive Normal Form (DNF). Formulas in $wff(L_O)$ can be partially grounded, if they use at least one free variable, or fully grounded if they use no free variables.

In this paper we intensively use the concept of variable substitution. We define a substitution instance $\Theta = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_i \leftarrow t_i\}$ as the substitution of the terms $t_1, t_2, \ldots, t_i$ for variables $x_1, x_2, \ldots, x_i$ in a formula $f \in wff(L_O)$.

We denote the set of roles in a normative system as the set of constants $R$, where $R \subset O$, and the set of participants as $P$, where each participant enacts at least one role according to the ontology $O$.

As our aim is to build a normative monitoring mechanism that can work at real time, special care has been made to choose a norm language which, without loss of expressiveness, has operational semantics that can then be mapped into production systems. Based in our previous work and experience, our definition of norm in an extension of the norm language presented in [12]:

**Definition 1.** A norm $n$ is a tuple $n = (f_A, f_M, f_D, f_w, w)$, where

- $f_A, f_M, f_D, f_w \in wff(L_O), w \in R$,
- $f_A, f_M, f_D$ respectively represent the activation, maintenance, and deactivation conditions of the norm, $f_w$ is the explicit representation of the deadline and target of the norm, and
- $w$ is the subject of the norm.

In order to create an optimal norm monitor it is important to know which norms are active at each point in time, as only those are the ones that have to be traced (inactive norms can be discarded from the monitoring process until they become active again). The activation condition $f_A$ specifies when a norm becomes active. It is also the main element in the norm instantiation process: when the conditions in the activating condition hold, the variables are instantiated, creating a new norm instance.$^3$

The target condition $f_w$ describes the state that fulfills the norm (e.g. if one is obliged to pay, the payment being made fulfills the obligation). The deactivating condition $f_D$ defines when the norm becomes inactive. Typically it corresponds to the target condition (e.g., fulfilling the norm instance deactivates that instance of the norm), but in some cases it also adds conditions to express other deactivating scenarios (e.g., when the norm becomes deprecated). The maintenance condition $f_M$ defines the conditions that, when

---

$^3$ One main differentiating aspect of our formalisation is that we include variables in the norm representation and we can handle multiple instantiations of the same norm and track them separately.
no longer hold, lead to a violation of the norm. Finally the deadline condition \( f_\delta \) represents one or several deadlines for the norm to be fulfilled.

An example of a norm for the traffic scenario ("A person driving on a street is not allowed to break a traffic convention") would be formalised as follows

\[
n_1 := \langle \text{enacts}(X, \text{Driver}) \land \text{is_driving}(X), \\
\neg \text{traffic_violation}(X), \\
\bot, \\
\neg \text{is_driving}(X), \\
\text{is_driving}(X) \land \neg \text{traffic_violation}(X), \\
\text{Driver} \rangle,
\]

The activating condition states that each time an event appears where an individual enacting the \text{Driver} role drives (\text{is_driving}), then a new instance of the norm becomes active; the maintenance condition states that the norm will not be violated while no traffic convention is violated; this norm has no deadline, it is to apply at all times an individual is driving; the norm instance deactivates when the individual stops driving\(^4\); the target of this norm is that we want drivers not breaking traffic conventions; finally the subject of the norm is someone enacting the \text{Driver} role.

It is important to note here that, although our norm representation does not explicitly include deontic operators, the combination of the activation, deactivation and maintenance conditions is as expressive as conditional deontic statements with deadlines as the ones in [3]. It is also able to express unconditional norms and maintenance obligations (i.e. the obligation to keep some conditions holding for a period of time). To show that our representation can be mapped to conditional deontic representations, let us express the semantics of the norm in definition 1 in terms of conditional deontic statements.

Given relations between the deadline and maintenance condition (that is, \( f_\delta \rightarrow \neg f_M \)), since the maintenance condition expresses more than the deadline alone) and between the target and the deactivation condition (i.e., \( f_w \rightarrow f_D \), since the deactivation condition specifies that either the norm is fulfilled or something special has happened), we can formalise the norms of definition 1 as the equivalent deontic expression (using the formalism of [3]): \( f_A \rightarrow [O_w(E_w f_w \leq \neg f_M) U f_D] \), where \( E_ap \) means that agent \( a \) sees to it that (\( stit \)) \( p \) becomes true and \( U \) is the CTL\(^*\) until operator. Intuitively, the expression states that after the norm activation, the subject is obliged to see to it that the target becomes true before the maintenance condition is negated (either the deadline is reached or some other condition is broken) until the norm is deactivated (which is either when the norm is fulfilled or has otherwise expired).

Since we are not reasoning about the (effects of) combinations of norms, we will not go into further semantical details here. The semantics presented in this deontic reduction are enough for understanding the monitoring process that is detailed in the remainder of the paper.

A set of norms is denoted as \( N \). We define as violation handling norms those norms that are activated automatically by the violation of another norm:

\(^4\) Although the norm is to apply at all times an individual is driving, it is better to deactivate the norm each time the individual stops driving, instead to keep it active, to minimize the number of norm instances the monitor needs to keep track at all times.
Definition 2. A norm \( n' = \langle f'_A, f'_M, f'_D, f'_w, w' \rangle \) is a violation handling norm of \( n = \langle f_A, f_M, f_D, f_w, w \rangle \), denoted as \( n \rightarrow n' \) iff \( f_A \land \neg f_M \land \neg f_D \equiv f'_A \).

Violation handling norms are special in the sense that they are only activated when another norm is violated. They are used as sanctioning norms, if they are to be fulfilled by the norm violating actor (e.g., the obligation to pay a fine if the driver broke a traffic sign), or as reparation norms, if they are to be fulfilled by an institutional actor (e.g. the obligation of the authorities to fix the broken traffic sign).

A norm is defined in an abstract manner, affecting all possible participants enacting a given role. Whenever a norm is active, we will say that there is a norm instance \( ni = (n, \theta) \) for a particular norm \( n \) and a substitution instance \( \Theta \).

We define the state of the world \( s_t \) at a specific point of time \( t \) as the set of predicates holding at that specific moment, where \( s_t \subseteq O \), and we will denote \( S \) as the set of all possible states of the world, where \( S = \mathcal{P}(O) \). We will call expansion \( F(s) \) of a state of the world \( s \) as the minimal subset of \( wff(\mathcal{L}_O) \) that uses the predicates in \( s \) in combination of the logical connectives \{\( \neg \), \( \lor \), \( \land \)\}.

One common problem for the monitoring of normative states is the need for an interpretation of brute events as institutional facts, also called constitution of social reality[8]. The use of counts-as rules helps solving this problem. Counts-as rules are multi-modal statements of the form \( [c]\gamma_1 \rightarrow \gamma_2 \), read as “in context \( c \), \( \gamma_1 \) counts-as \( \gamma_2 \)”. In this paper, we will consider a context as a set of predicates, that is, as a possible subset of a state of the world:

Definition 3. A counts-as rule is a tuple \( c = \langle \gamma_1, \gamma_2, s \rangle \), where \( \gamma_1, \gamma_2 \in wff(\mathcal{L}_O) \), and \( s \subseteq O \).

A set of counts-as rules is denoted as \( C \). Although the definition of counts-as in [8] assumes that both \( \gamma_1 \) and \( \gamma_2 \) can be any possible formula, in our work we limit \( \gamma_2 \) to a conjunction of predicates for practical purposes.

Definition 4. Following the definitions above, we define an institution as a tuple of norms, roles, participants, counts-as rules, and an ontology:

\[ I = \langle N, R, P, C, O \rangle \]

An example of \( I \) for the traffic scenario would be formalised as follows:

\[ N := \{\langle enacts(X, Driver) \land is\_driving(X), \neg traffic\_violation(X), \bot, \neg is\_driving(X), is\_driving(X) \land \neg traffic\_violation(X), Driver\rangle, \langle enacts(X, Driver) \land is\_driving(X) \land traffic\_violation(X), \top, \neg, paid\_fine(X), Driver\rangle \} \]

\[ R := \{Driver\}, P := \{Person_1\} \]

\[ C := \{\langle exceeds(D, 50), traffic\_violation(D), is\_in\_city(D)\rangle \} \]

\[ O := \{role, enacts, is\_driving, is\_in\_city, exceeds, traffic\_violation, is\_driving, paid\_fine, Person_1, role(Driver), enacts(Person_1, Driver)\} \]
2.2 Normative Monitor

In this section we present a formalisation of normative monitoring based on Structural Operational Semantics.

From the definitions introduced in section 2.1, a Normative Monitor will be composed of the institutional specification, including norms, the current state of the world, and the current normative state.

In order to track the normative state of an institution at any given point of time, we will define three sets: an instantiation set $IS$, a fulfillment set $FS$, and a violation set $VS$, each of them containing norm instances $\{\langle n, \Theta \rangle, \langle n', \Theta' \rangle, \ldots, \langle n'', \Theta'' \rangle \}$. We adapt the semantics for normative states from [11]:

Definition 5. Norm Lifecycle: Let $n_i = \langle n, \Theta \rangle$ be a norm instance, where $n = \langle f_A, f_M, f_D, w \rangle$, and a state of the world $s$ with an expansion $F(s)$. The lifecycle for norm instance $n_i$ is defined by the following normative state predicates:

- $\text{activated}(n_i) \iff \exists f \in F(s), \Theta(f_A) \equiv f$
- $\text{maintained}(n_i) \iff \exists f' \in F(s), \Theta'(f_M) \equiv f \land \Theta' \subseteq \Theta$
- $\text{deactivated}(n_i) \iff \exists f' \in F(s), \Theta'(f_D) \equiv f \land \Theta' \subseteq \Theta$
- $\text{instantiated}(n_i) \iff n_i \in IS$
- $\text{violated}(n_i) \iff n_i \in VS$
- $\text{fulfilled}(n_i) \iff n_i \in FS$

where $IS$ is the instantiation set, $FS$ is the fulfillment set, and $VS$ is the violation set, as defined above.

For instance, for norm $n_1$, the lifecycle is represented in Figure 1. The maintained state is not represented as it holds in both the activated and fulfilled states. The deactivated state is also not depicted because it corresponds in this case to the Fulfilled state.

![Fig. 1: Lifecycle for norm n1 in the traffic scenario: (I)nactive, (A)ctivated, (V)iolated, (F)ulfilled](image)

Definition 6. A Normative Monitor $M_I$ for an institution $I$ is a tuple $M_I = \langle I, S, IS, VS, FS \rangle$ where
\[ I = \langle N, R, P, C, O \rangle, \]
\[ S = \mathcal{P}(O), \]
\[ IS = \mathcal{P}(N \times S \times \text{Dom}(S)), \]
\[ VS = \mathcal{P}(N \times S \times \text{Dom}(S)), \]
\[ FS = \mathcal{P}(N \times S \times \text{Dom}(S)). \]

The set \( \Gamma \) of possible configurations of a Normative Monitor \( M_I \) is
\[ \Gamma = I \times S \times IS \times VS \times FS. \]

However, the definition above does not take into account the dynamic aspects of incoming events affecting the state of the world through time. To extend our model we will assume that there is a continuous, sequential stream of events received by the monitor:

**Definition 7.** An event \( e \) is a tuple \( e = \langle \alpha, p \rangle \), where

\[ - \quad \alpha \in I^5 \]
\[ - \quad p \in F \text{ and is fully grounded.} \]

We define \( E \) as the set of all possible events, \( E = \mathcal{P}(P \times F) \)

**Definition 8.** The Labelled Transition System for a Normative Monitor \( M_I \) is defined by \( \langle \Gamma, E, \triangleright \rangle \) where

\[ - \quad E \text{ is the set of all possible events } e = \langle \alpha, p \rangle \]
\[ - \quad \triangleright \text{ is a transition relation such that } \triangleright \subseteq \Gamma \times E \times \Gamma \]

The inference rules for the transition relation \( \triangleright \) are depicted in Figure 2.

### 3 Monitoring with production systems

In our approach, practical normative reasoning is based on a production system with an initial set of rules implementing the operational semantics described in Section 2.2. Production systems are composed of a set of rules, a working memory, and a rule interpreter or engine [2]. Rules are simple conditional statements, usually of the form

\[ \text{IF } a \text{ THEN } b, \]

where \( a \) is usually called left-hand side (LHS) and \( b \) is usually called right-hand side (RHS).

#### 3.1 Semantics of production systems

In this paper we use a simplified version of the semantics for production systems introduced in [1].

Considering a set \( \mathcal{P} \) of predicate symbols, and an infinite set of variables \( \mathcal{X} \), where a fact is a ground term, \( f \in \mathcal{T}(\mathcal{P}) \), and \( WM \) is the working memory, a set of facts, a production rule is denoted if \( p, c \) remove \( r \) add \( a \), or

\[ \text{\( p \) is considered to be the asserter of the event. Although we are not going to use this element in this paper, its use may be of importance when extending or updating this model.} \]
Event processed:

\[ e_i = (\alpha, p) \]
\[ \langle (\alpha, p), e_i, e_{i+1} \rangle \]  

Counts-as rule activation:

\[ \exists \Theta, \exists f \in F(s), \exists (\gamma_1, \gamma_2, s_i) \in C, s_i \subseteq s \land \Theta(\gamma_1) \equiv f \land \Theta(\gamma_2) \notin s \]
\[ \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \]

Counts-as rule deactivation:

\[ \exists \Theta, \exists f \in F(s), \exists (\gamma_1, \gamma_2, s_i) \in C, s_i \nsubseteq s \land \Theta(\gamma_1) \equiv f \land \Theta(\gamma_2) \in s \]
\[ \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \]

Norm instantiation:

\[ \exists n = \left\{ (fA, fM, fD, w) \in N \land \neg \exists n' \in N, n' \rightarrow n \land (n, \Theta) \notin is \land \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(fA) \rightarrow \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \right\} \]

Norm instance violation:

\[ \exists n = \left\{ (fA, fM, fD, w) \in N \land (n, \Theta') \in is \land \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(fA) \rightarrow \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \right\} \]

Norm instance fulfilled:

\[ \exists n = \left\{ (fA, fM, fD, w) \in N \land (n, \Theta') \in is \land \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(fA) \rightarrow \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \right\} \]

Norm instance violation repaired:

\[ \exists n, n' \in N \land n \rightarrow n' \land (n, \Theta) \in is \land \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(fA) \rightarrow \langle (N, R, P, C, O), s, is, vs, fs, e \rangle \]

Fig. 2: Inference rules for the transition relation $\triangleright$

\[ p_r, c \Rightarrow r, a, \]

consisting of the following components:

- A set of positive or negative patterns $p = p^+ \cup p^-$ where a pattern is a term $p_i \in \mathcal{T}(\mathcal{F}, X)$ and a negated pattern is denoted $\neg p_i$. $p^+$ is the set of all negated patterns and $p^+$ is the set of the remaining patterns.
- A proposition $c$ whose set of free variables is a subset of the pattern variables: $\text{Var}(c) \subseteq \text{Var}(p)$.
- A set $r$ of terms whose instances could be intuitively considered as intended to be removed from the working memory when the rule is fired, $r = \{ r_i \}_{i \in I_r}$, where $\text{Var}(r) \subseteq \text{Var}(p^+)$. 

A set of terms whose instances could be intuitively considered as intended to be added to the working memory when the rule is fired, \( a = \{a_i\}_{i \in I_a} \), where \( \text{Var}(a) \subseteq \text{Var}(p) \).

**Definition 9.** A set of positive patterns \( p^+ \) matches to a set of facts \( S \) and a substitution \( \sigma \) iff \( \forall p \in p^+, \exists t \in S, \sigma(p) = t \). Similarly, a set of negative patterns \( p^- \) dismatches a set of facts \( S \) iff \( \forall \neg p \in p^-, \forall t \in S, \forall \sigma, \sigma(p) \neq t \).

A production rule \( p \Rightarrow r, a \) is \((\sigma, \text{WM}')\)-fireable on a working memory \( \text{WM} \) when \( p^+ \) matches with \( \text{WM}' \) and \( p^- \) dismatches with \( \text{WM} \), where \( \text{WM}' \) is a minimal subset of \( \text{WM} \), and \( T \models \sigma(c) \).

**Definition 10.** The application of a \((\sigma, \text{WM}')\)-fireable rule on a working memory \( \text{WM} \) leads to the new working memory \( \text{WM}'' = (\text{WM} - \sigma(r)) \cup \sigma(a) \).

**Definition 11.** A general production system \( \mathcal{PS} \) is defined as \( \mathcal{PS} = \langle \mathcal{P}, \text{WM}_0, \mathcal{R} \rangle \), where \( \mathcal{R} \) is a set of production rules over \( \mathcal{H} = \langle \mathcal{P}, X \rangle \).

### 3.2 Reduction

In order to formalise our Normative Monitor as a production system, we will need to define several predicates to bind norms to their conditions: activation, maintenance, deactivation, and to represent normative state over norm instances: violated, instantiated, and fulfilled. We will also use a predicate for the arrival of events: event, and a predicate to represent the fact that a norm instance is a violation handling norm instance of a violated instance: repair. For the handling of the DNF clauses, we will use the predicates holds and has\_clause.

**Definition 12.** The set of predicates for our production system, for an institution \( I = \langle N, R, P, C, O \rangle \), is:

\[ \mathcal{P}_I := O \cup \{\text{activated}, \text{maintained}, \text{deactivated}, \text{violated}, \text{instantiated}, \text{fulfilled}, \text{event}, \text{repair}, \text{holds}, \text{has\_clause}, \text{countsas}\} \]

The initial working memory \( \text{WM}_0 \) should include the institutional specification in the form of the formulas included in the counts-as rules and the norms in order to represent the possible instantiations of the predicate holds, through the use of the predicate has\_clause.

First of all, we need to have the bindings between the norms and their formulas available in the working memory. For each norm \( n = \langle f_A, f_M, f_D, w \rangle \), these bindings will be:

\[ \text{WM}_n := \{\text{activation}(n, f_A), \text{maintenance}(n, f_M), \text{deactivation}(n, f_D)\} \]

As we assume the formulas from \( \text{wff}(\mathcal{C}_O) \) to be in DNF form:

**Definition 13.** We can interpret a formula as a set of conjunctive clauses \( f = \{f_1, f_2, ..., f_N\} \), of which only one of these clauses \( f_i \) holding true is necessary for \( f \) holding true as well:

\[ r^h := \text{has\_clause}(f, f') \land \text{holds}(f', \Theta) \Rightarrow \emptyset, \{\text{holds}(f, \Theta)\} \]
Definition 14. The initial working memory \( WM_I \) for an institution \( I = \langle N, R, P, C, O \rangle \) is:

\[
WM_I := \bigcup_{n \in N} repair(n, n') \cup \bigcup_{n=(f_a,f_m,f_d,w) \in N} (WM_n \cup WM_{f_a} \cup WM_{f_m} \cup WM_{f_d}) \cup \bigcup_{c=(\gamma_1,\gamma_2,s) \in C} (\{\text{countsas}(\gamma_1, \gamma_2, s)\} \cup WM_{\gamma_1} \cup WM_s)
\]

The rule for the detection of a holding formula is defined as \( r^h_{f,c} = [f] \Rightarrow \emptyset, \{\text{holds}(f, \sigma)\} \), where we denote as \( [f] \) the propositional content of a formula \( f \in w f f(L_O) \) which only uses predicates from \( O \) and the logical connectives \( \neg \) and \( \land \), and \( \sigma \) as the substitution set of the activation of the rule. Following the previous example:

\[
r^h_{f_1,c} = p_1(x) \land p_2(y) \land \ldots \land p_i(z) \Rightarrow \emptyset, \{\text{holds}(f_1, \{x, y, z\})\}
\]

\[
r^h_{f_2,c} = q_1(w) \land q_2(x) \land \ldots \land q_i(y) \Rightarrow \emptyset, \{\text{holds}(f_2, \{w, x, y\})\}
\]

Similarly as in Definition 14:

Definition 15. The set of rules \( R^h_{f,c} \) for detection of holding formulas for an institution \( I = \langle N, R, P, C, O \rangle \) is:

\[
R^h_{f,c} := \bigcup_{n=(f_a,f_m,f_d,w) \in N} (\bigcup_{f \in \{f_a,f_m,f_d\}} r^h_{f,c}) \cup \bigcup_{c=(\gamma_1,\gamma_2,s) \in C} (\bigcup_{f \in \gamma_1} r^h_{f,c})
\]

By using the predicate \text{holds} as defined above, we can translate the inference rules from Section 2.2. Please note that the rules are of the form \( p, c \Rightarrow r, a \) as shown in Section 3.1. However, as we only need the \( c \) part to create a constraint proposition in the rules for norm instance violation and fulfillment, \( c \) is omitted except for these two particular cases.

Definition 16. Translated rules (see Figure 2)

Rule for event processing [1]:

\[
r^e = \text{event}(\alpha, p) \Rightarrow \emptyset, \{|p|\}
\]

Rule for counts-as rule activation [2]:

\[
r^{ca} = \text{countsas}(\gamma_1, \gamma_2, c) \land \text{holds}(\gamma_1, \Theta) \land \text{holds}(c, \Theta') \land \neg\text{holds}(\gamma_2, \Theta)
\]

\[
\Rightarrow \emptyset, \{\Theta(\{\gamma_2\})\}
\]

Rule for counts-as rule deactivation [2]:

\[
r^{cd} = \text{countsas}(\gamma_1, \gamma_2, c) \land \text{holds}(\gamma_1, \Theta) \land \neg\text{holds}(c, \Theta') \land \text{holds}(\gamma_2, \Theta)
\]

\[
\Rightarrow \{\Theta(\{\gamma_2\})\}, \emptyset
\]

Rule for norm instantiation [2]:

\[
r^{ni} = \text{activation}(n, f) \land \text{holds}(f, \Theta) \land \neg\text{instantiated}(n, \Theta) \land \neg\text{repair}(n', n)
\]

\[
\Rightarrow \emptyset, \{\text{instantiated}(n, \Theta)\}
\]

Rule for norm instance violation [5]:

\[
r^{vi} = \text{instantiated}(n, \Theta) \land \text{maintenance}(n, f) \land \neg\text{holds}(f, \Theta') \land \text{repair}(n', n),
\]

\[
\forall \Theta' \subseteq \Theta
\]
Rule for norm instance fulfillment (6):
\[ r_{nf} = \text{deactivation}(n, f) \land \text{instantiated}(n, \Theta) \land \subseteq(\Theta, \Theta') \land \text{holds}(f, \Theta') \]
⇒ \{\text{instantiated}(n, \Theta)\}, \{\text{fulfilled}(n, \Theta)\}

Rule for norm instance violation repaired (7):
\[ r_{nr} = \text{violated}(n, \Theta) \land \text{repair}(n, n', \Theta') \land \text{fulfilled}(n', \Theta') \]
⇒ \{\text{violated}(n, \Theta)\}, \emptyset

Definition 17. Following Definitions 13, 15 and 16, the set of rules for an institution \( I = \langle N, R, P, C, O \rangle \) are:
\[ R_I := R_{hc}^{I} \cup \{r_h, r_e, r_{ax}, r_{ed}, r_{ni}, r_{ne}, r_{nf}, r_{nr}\} \]

Definition 18. The production system \( PS_I \) for an institution \( I \) will be, from Definitions 12, 14, and 17:
\[ PS_I := \langle P_I, WM_I, R_I \rangle \]

4 Implementation

A prototype of our normative reasoner has been implemented as a DROOLS program. DROOLS is an open-source Object-Oriented rule engine for declarative reasoning in Java [14]. Its rule engine is an implementation of the forward chaining inference Rete algorithm [3]. The use of Java objects inside the rule engine allows for portability and an easier communication of concepts with the reasoning of agents coded in Java.
rule "holds" when HasClause(f : formula, f2 : clause) Holds(formula == f2, theta : substitution) then insertLogical(new Holds(f, theta)); end

rule "event processed" when Event(a : asserter, p : content) then insertLogical(p); end

rule "counts-as activation" when CountsAs(g1 : gamma1, g2 : gamma2, s : context) Holds(formula == g1, theta : substitution) not Holds(formula == g2, substitution == theta) then Formula f;
  f = g2.substitute(theta);
  insert(f); end

rule "counts-as deactivation" when CountsAs(g1 : gamma1, g2 : gamma2, s : context) Holds(formula == g1, theta : substitution) not Holds(formula == g2, substitution == theta) f : Formula(content == g2, grounding == theta) then retract(f); end

rule "norm instantiation" when Activation(n : norm, f : formula) Holds(formula == f, theta : substitution) not Instantiated(norm == n, substitution == theta) not Repair(n2 : norm, repairNorm == n) then insert(new Instantiated(n, theta)); end

rule "norm instance violation" when ni : Instantiated(n : norm, theta : substitution) Maintenance(norm == n, f : formula) not (SubsetEQ(theta2 : subset, superset == theta) and Holds(formula == f, substitution == theta2)) Repair(norm == n, n2 : repairNorm) then retract[ni];
  insert(new Violated[n, theta]);
  insert(new Instantiated(n2, theta2)); end

rule "norm instance fulfillment" when Deactivation(n : norm, f : formula) ni : Instantiated(norm == n, theta : substitution) SubsetEq(theta2 : subset, superset == theta) Holds(formula == f, substitution == theta2) then retract[ni];
  insert(new Fulfilled(n, theta)); end

rule "norm instance violation repaired" when ni : Violated(n : norm, theta : substitution) Repair(norm == n, n2 : repairNorm) Fulfilled(norm == n2, substitution == theta) then retract[ni]; end

rule "norm instantiation" when Activation(n : norm, f : formula) Holds(formula == f, theta : substitution) not Instantiated(norm == n, substitution == theta) not Repair(n2 : norm, repairNorm == n) then insert(new Instantiated(n, theta)); end

rule "subseteq" when Holds(f : formula, theta : substitution) Holds(f2 : formula, theta2 : substitution) eval(theta.containsAll(theta2)) then insertLogical(new SubsetEQ(theta2, theta)); end

Fig. 4: Translation of base rules to DROOLS
In DROOLS we can represent facts by adding them to the knowledge base as objects of the class Predicate. Predicates are dynamically imported from standardised Description Logic OWL-DL ontologies into Java objects using the tool OWL2Java\cite{17}, as subclasses of a specifically designed Predicate class. The following shows an example of the insertion of \textit{enacts\_role}(p, Driver) into the knowledge base to express that p (represented as object p of the domain and instantiating a participant) is in fact enacting the role \textit{driver}:

\begin{verbatim}
ksession.insert(new Enacts(p, Driver.class));
\end{verbatim}

DROOLS programs can be initialised with a rule definition file. However, its working memory and rule base can be modified at run-time by the Java process that is running the rule engine. We take advantage of this by keeping a fixed base, which is a file with fixed contents implementing the rules from Definition \cite{13} and \cite{16} which are independent of the institution, and having a parser for institutional definitions that will feed the rules from Definition \cite{15} which are dependent on the institution (see Figure 3). The institutional definitions we currently use are based on an extension of the XML language presented in \cite{12}.

The base rules (see Definitions \cite{13} and \cite{16}) has been quite straightforward and the translation is almost literal. The contents of the reusable DROOLS file is shown in Figure 4. The last rule of the Figure is the declarative implementation of the predicate \textit{SubsetEQ} to represent the comparison of substitutions instances $\Theta \subseteq \Theta'$, needed for the cases of norm instance violation and fulfillment. In our implementation in DROOLS, substitution instances are implemented as Set\(<\text{Value}>\) objects, where Value is a tuple \langle String, Object \rangle.

The rest of the rules (see Definitions \cite{15}) are automatically generated from the institutional specifications and inserted into the DROOLS rule engine. An example of two generated rules for the traffic scenario is shown in Figure 5.

\begin{verbatim}
rule "N1\_activation\_1"
when
  n : Norm(id == "N1")
  Activation(norm == n, f : formula)
  Enacts(X : p0, p1 == "Driver")
  IsDriving(p0 == X)
then
  Set\<Value\> theta = new Set\<Value\>();
  theta.add(new Value("X", X));
  insert(new Holds(f.getClause(0), theta));
end

rule "C1\_1"
when
  c : CountsAs(g1 : gamma1)
  Exceeds(D : p0, 50 : p1)
then
  Set\<Value\> theta = new Set\<Value\>();
  theta.add(new Value("D", D));
  insert(new Holds(g1.getClause(0), theta));
end
\end{verbatim}

Fig. 5: Rules for the traffic scenario
The initial working memory is also automatically generated by inserting objects (facts) into the DROOLS knowledge base following Definition 14. An example for the traffic scenario is also shown in Figure 6. Please note that this is not an output of the parser, but a representation of what it would execute at run-time.

```java
ksession.insert(norm1);
ksession.insert(norm2);
ksession.insert(new Repair(norm1, norm2));
ksession.insert(new Activation(norm1, fn1a));
ksession.insert(new Maintenance(norm1, fn1m));
ksession.insert(new Deactivation(norm1, fn1d));
ksession.insert(new HasClause(fn1a, fn1a1));
ksession.insert(new HasClause(fn1m, fn1m1));
ksession.insert(new HasClause(fn1d, fn1d1));
/* ...same for norm2... */
ksession.insert(new CountsAs(c1g1, c1g2, c1s));
ksession.insert(new HasClause(c1g1, c1g11));
ksession.insert(new HasClause(c1g2, c1g21));
ksession.insert(new HasClause(c1s, c1s1));
```

Fig. 6: Facts for the traffic scenario

5 Conclusions and Related Work

The implementation of rule-based norm operationalisation has already been explored in previous research. Some approaches [13,15] directly define the operationalisation of the norms as rules of a specific language, not allowing enough abstraction to define norms at a high level to be operationalised in different rule engine specifications. [5] introduces a translation scheme, but it is bound to Jess by using specific constructs of this language and it does not support constitutive norms.

Other recent approaches like [6] define rule-based languages with expressive constructs to model norms, but they are bound to a proper interpreter and have no grounding on a general production system, requiring the use of an intentionally crafted or modified rule engine. For example, in [7,9], obligations, permissions and prohibitions are asserted as facts by the execution of the rules, but the actual monitoring is out of the base rule engine used.

[16] introduces a language for defining an organisation in terms of roles, norms, and sanctions. This language is presented along with an operational semantics based on transition rules, thus making its adoption by a general production system straightforward. Although a combination of counts-as rules and sanctions is used in this language, it is not expressive enough to support regulative norms with conditional deontic statements.

We solve these issues by combining a normative language [12] with a reduction to a representation with clear operational semantics based on the framework in [11] for deontic norms and the use of counts-as rules for constitutive norms. The formalism presented in this paper uses logic conditions that determine the state of a norm (active,
fulfilled, violated). These conditions can be expressed in propositional logic and can be directly translated into LHS parts of rules, with no special adaptation needed. The implementation of the operational semantics in a production system to get a practical normative reasoner is thus straightforward. This allows agents for dynamically changing its institutional context at any moment, by feeding the production system with a new abstract institutional specification.

Our intention is not to design a general purpose reasoner for normative agents, but a practical reasoner for detecting event-driven normative states. This practical reasoner can then be used as a component not only by normative agents, but also by monitors or managers. Normative agents should deal with issues such as planning and future possibilities, but monitors are focused on past events. For such a practical reasoner, the expressivity of actions languages like C+ is not needed, and a simple yet efficient solution is to use production systems, as opposed to approaches more directly related to offline verification or model checking, such as [10].

Mere syntactical translations are usually misleading in the sense that rule language specific constructs are commonly used, constraining reusability [13,57]. However, as we have presented in this paper a reduction to a general version of production system semantics, any rule engine could fit our purposes. There are several production system implementations available, some widely used by the industry, such as JESS, DROOLS, SOAR or PROVA. In most of these systems rules are syntactically and semantically similar, so switching from one to the other would be quite simple. As production systems dynamically compile rules to efficient structures, they can be used as well to validate and verify the consistency of the norms. As opposed to [79], our reduction ensures that the whole monitoring process is carried out entirely by a general production system, thus effectively decoupling normative state detection and agent reasoning.

DROOLS is an open-source powerful suite supported by JBoss, the community, and the industry, and at the same time it is lightweight enough while including key features that we are or will be using in future work. As an advantage over other alternatives, it includes features relevant to our topic, e.g. event processing, workflow integration. Its OO approach makes it easy to be integrated with imperative code (Java), and OWL-DL native support is expected in a short time.

Our implementation is already being used in several use cases with large amounts of events and it is available at [http://ict-alive.svn.sf.net/viewvc/ict-alive/OrganisationLevel/trunk/Monitoring/] under a GPL license. As future work we expect to extend the semantics in order to support first-order logic norm conditions, and to perform an analysis on the algorithmic complexity of our implementation.

Acknowledgements

This work has been partially supported by the FP7-215890 ALIVE project. J. Vázquez-Salceda’s work has been also partially funded by the Ramón y Cajal program of the Spanish Ministry of Education and Science.
References


Controlling multi-party interaction within normative multi-agent organizations

Olivier Boissier¹, Flavien Balbo²,³, and Fabien Badeig²,³

¹ Ecole Nationale Supérieure des Mines, 158 Cours Fauriel, 42100 Saint-Etienne, France
  Olivier.Boissier@emse.fr
² Université Paris-Dauphine - LAMSADE, Place du Maréchal De Lattre de Tassigny, F-75775 Paris 16 Cedex, France
  balbo@lamsade.dauphine.fr
³ INRETS - GRETIA, 2, Rue de la Butte Verte, 93166 Noisy Le Grand, France
  badeig@inrets.fr

Abstract. Multi-party communications taking place within organizations lead to different interaction modes between agents (e.g. (in)direct communication between roles, (in)direct communication restricted to a group, etc). Fully normative organisations need to regulate and control these modes as they do for agents' behaviors. This paper proposes to extend the normative organization model \( \text{Moose} \) in order to specify such interaction modes between autonomous agents participating to an organization. This specification has two purposes: (i) to make the multi-agent organization able to monitor the interaction between the agents, (ii) to make the agents able to reason on these modes as they can do on norms. The paper is focused on the first point. We illustrate with a crisis management application how this extension has been implemented thanks to a specialization of the Easi interaction model.

1 Introduction

In a Multi-Agent System (MAS), interaction and organization play key and essential roles. A MAS is often described as composed of agents situated in a shared environment interact directly or indirectly with each other to execute and cooperate in a distributed and decentralized setting according to an organization. The Easi model⁴ [7] proposes a multi-party environment based interaction model and is therefore able to support the complexity of the interaction within an organization. On one hand, agents are able to send messages to other agents situated in the environment and, on the other hand, any agent situated in the environment is able to perceive the exchanged message. It is thus possible to consider, for the same message, direct, indirect and overhearing communications. If needed, the Easi model preserves the privacy of the interaction, more details are given in [7].

⁴ Environment as Active Support for Interaction
However, it is not possible to make the agents aware of this interaction settings. There is no declarative representation usable at the agent level.

Agents in a MAS are often structured along one organization that helps and/or constrains their cooperation schemes. Current proposals offer modeling languages usable either by agents either by an organization management system dedicated to the regulation and supervision of the agents within the defined organization. The Moise model [5] is one of these proposals. Its organization modeling language is composed of two dimensions – structural and functional - connected to each other by a normative specification. Such a feature makes it possible to easily extend the model with new dimensions. Currently, there doesn’t exist any dimension dedicated to the definition of interaction modes within the organization. It is thus not possible to govern the agents interaction modes resulting from the multi-party communications offered in the Easi framework.

In this paper, our aim is to propose a unified model for interaction and organization. To reach this objective we propose the enrichment of the Moise organization modeling language with a new and independent dimension connected to the other ones by the normative specification. This way it is thus possible to make the agents able of reasoning on their use of the interaction modes offered in the Easi platform. The next step will be to develop such reasoning capabilities in the agents. In this paper we focus on the presentation of the unified model and how it is translated to be monitored by the facilities offered by the Easi platform. The MAS designer will be able to use the resulting specification of both the organization and the interaction and to get the corresponding support environment.

The paper is organized as follows. In section 2, we present the background of the proposal and motivate our choices. In section 3, we expose how Moise organization modeling language has been extended to specify the interaction modes proposed by Easi. The section 4 describes how this specification is mapped to the Easi model. In section 5, we show the expressing capabilities of the proposal with different examples issued of a crisis management application. Before conclusion, we compare our proposal to the current related approaches.

2 Background

In this paper, we consider a crisis management application where different dedicated emergency services must be coordinated in order to solve a crisis situation. The main difficulty in the modeling of such an application consists in the definition of the interaction constraints between those services, given that each service has the possibility to decide on its own which interaction mode to use. We use this application all along the paper to illustrate the components of our proposal. In the following sections, we specify parts of the multi-agent organization governing this application thanks to the models Moise and Easi.
2.1 Moise

The Moise framework [4] is composed of an organization modeling language, an organization management infrastructure and organization based reasoning mechanisms at the agent level. In this paper, we focus on the organization modeling language. Our aim is to use it with the EASI platform in order to specify and regulate the different interaction modes available on this platform (see next section).

The organization modeling language considers the specification of an organization along three independent dimensions\(^5\): structural (SS), functional (FS) and normative (NS). Whereas SS and FS are independent, NS defines a set of norms binding elements of both specifications. The aim is that the agents enact the behaviors specified in NS when participating to the organization. The organization modeling language is accompanied by a graphical language (cf. Fig. 1, 2) and XML is used to store the organizational specifications.

**Structural Dimension:** The structural dimension specifies the roles, groups, and links of an organization. It is defined with the following tuple: \((R, \sqsubset, rg)\) with \(R\) set of the roles, \(\sqsubset\) inheritance relation between roles, \(rg\) organization root group specification. The definition of this group gives the compatibility relations between roles, the maximal and minimal cardinality of agents that can endorse roles within the group, the links connecting roles to each other (communication, authority, acquaintance) and sub-groups. In NS, the role is used to bind a set of constraints on behaviors that the agent commits to satisfy as soon it endorses the role.

In the crisis application, we define (cf. Fig. 1) two main groups which correspond to the tactical spheres used in a crisis management: decision-making sphere (Decision-making) and operational sphere (Operational). For each of them, we define the roles manager and operator inheriting the generic role role-player. These roles are specialized respectively in coordinator, leader\(_D\) for the group Decision-making and leader\(_S\) for the subgroups of group Operational. The role coordinator (resp. leader\(_D\)) can be played by only and only one agent - 1.1 - (resp. several agents - 1.* -). A compatibility link connects the role leader\(_D\) to leader\(_S\) meaning that any agent playing leader\(_D\) will be able to play also the role leader\(_S\). Six communication links (cf. \(l_1\) to \(l_6\)) have been defined between these roles (e.g. \(l_1\) communication link between coordinator and leader\(_D\)).

**Functional Dimension:** The functional dimension is defined by \((M, G, S)\) with \(M\) set of missions, consistent grouping of collective or individual goals. A mission defines all the goals an agent commits to when participating in the execution of a social scheme by the way of the roles that they endorse. \(G\) is the set of the collective or individual goals to be satisfied and \(S\) is the set of social schemes, tree-like structurations of the goals into plans.

\(^5\) In this paper, we will provide the only necessary details in order to globally understand the model as well as the proposed extensions. For further details, readers should refer to http://moise.sourceforge.net/.
The Fig. 2 illustrates a social scheme of $FS$ expressing the collective plan for deciding within the crisis management application. According to it, agents should aggregate the different information in relation to the crisis situation Refining crisis perception, Safeguarding zone by executing one of the two social schemes (scheme 1 or scheme 2 that are not detailed here) and execute the scheme 3. The different goals are organized into missions.

**Normative Dimension:** The normative dimension $NS$ defines a set of norms as: $(id, c, \rho, dm, m)$ with $id$ norm identifier, $c$ activation condition of the norm, $\rho$ role concerned by the deontic modality, $dm$ deontic modality (obligation or permission), $m$ mission. A normative expression can be read as: “when $c$ holds, any agent playing role $\rho$ has $dm$ to commit on the mission $m$”. Within this language, norms are either a *permission*, either an *obligation* for a role to commit to a mission. Goals are indirectly connected to roles since a mission is a set of goals. Interdictions are supposed to exist by default: if the normative specification doesn’t have any permission or obligation for a couple mission, role, any agent playing the role is forbidden to commit to mission. A norm becomes in the active state (resp. inactive) as soon as the condition $c$ holds (resp. doesn’t

---

6 Predicates bearing on the current organization state (e.g. plays, committed, etc) and/or bearing on particular configurations of the application.
hold). When the norm is active, the deontic expression attached to the norm can be verified. The norm can thus becomes fulfilled or unfulfilled.

For instance, in the crisis management application, the norm obliging agents playing the role leaderS in the group Traffic Network Management (TNM) to safeguard the zone where the accident took place (mission m4) is: \( \langle n_1, c_1, \text{leader}_S, \text{obligation}, m_4 \rangle \) where \( c_1 \) is \text{plays}(bearer, \text{leader}_S, \text{TNM})\). The term bearer refers to the agent that will play the role “bearer” in the context of the obligation issued from the instantiation of the norm in the organization entity (see below) and \text{plays} is a predicate satisfied when the agent plays the leaderS in an instance of group TNM. When the zone is secured, the agents playing the same role within the context of the group Rescue Unity (RU) deploys the intervening scheme (mission m5) following the norm: \( \langle n_2, c_2, \text{leader}_D, \text{obligation}, m_5 \rangle \) where \( c_2 \) is \text{plays}(bearer, \text{leader}_S, \text{RU})\).

Organizational Entity: An organizational entity (OE) is defined from the organizational specification \( OS \) and a set of agents \( A \) by the following tuple: \( \langle OS, A, G_I, SI, O \rangle \) where \( G_I \) is the set of concrete groups of the organization, i.e. groups dynamically created from the group specification of the \( OS \), \( SI \) is the set of concrete social schemes dynamically created in the \( OE \) from the social schemes specification in the \( OS \) and \( O \) is the set of obligations issued from the norms \( NS \) attached to agents of \( A \) whose conditions are satisfied [6].

2.2 Easi

The multi-party interaction model EASI supports the management of direct, indirect and overhearing communications in a MAS [7]. For cognitive agents, the common point between all these communication modes consists in the routing of the messages by identifying which agent should obtain which message and in which context. Solving this problem requires taking into account both sides of the sender and potential receivers. To this aim, EASI manages meta-informations on the MAS (agents, messages, context) in the communication environment and use them to help the agents to interact. The interaction model EASI is thus
defined by \((\Omega, \mathcal{D}, P, \mathcal{F})\) with \(\Omega = \{\omega_1, \ldots, \omega_m\}\) the set of entities \((\mathcal{A} \subset \Omega \text{ and } MSG \subset \Omega - \mathcal{A}\) set of agents and \(MSG\) set of messages -), \(\mathcal{D} = \{d_1, \ldots, d_m\}\) set of domain descriptions of the properties, \(P = \{p_1, \ldots, p_n\}\) set of properties, and \(\mathcal{F} = \{f_1, \ldots, f_k\}\) set of filters.

**Entity**: The entities are the meta-information on the MAS that EASI manages. An entity \(\omega_i \in \Omega\) is defined by \((e_r, e_d)\) where \(e_r\) is a reference to an element of the MAS and \(e_d\) is the description of that element. An element of the MAS can be agents \((\mathcal{A})\), messages \((MSG)\) and a reference is its physical address on the platform or other objects such as URL, mailbox, .... The description \(e_d\) is defined by a set of couples \((p_i, v_j)\) where \(p_i \in P\) and \(v_j\) is the value of the property for this entity. Any agent of the MAS has its own processing and knowledge settings. It is connected to the communication environment by the way of its description that it stores and updates in this environment. This description \(e_d\) is used for the routing of the informations to the reference \(e_r\).

**Property**: A property gives an information on an entity. A property \(p_i \in P : \Omega \rightarrow d_j \cup \{\text{unknown}, \text{null}\}\) is a function whose description domain \(d_j \in \mathcal{D}\) can be quantitative, qualitative or a finite set of data. The \(\text{unknown}\) value is used when the value of the property cannot be set, and \(\text{null}\) is used to state that the property is undefined in the given description. In order to simplify the notation, only the value of the description domain is given to specify a property.

For instance, in the crisis management application, the properties attached to agents and messages are \(id, role, position, subject, sender\) with:

\[
\begin{align*}
- id &: \Omega \rightarrow N, \\
- role &: \Omega \rightarrow \{\text{coordinator, leader}\}, \\
- position &: \Omega \rightarrow N \times N, \\
- subject &: \Omega \rightarrow \{\text{alert, demand}\}, \\
- sender &: \Omega \rightarrow N.
\end{align*}
\]

An agent \(a\) can have the following description \(\{\text{role, coordinator}\}\) and an agent \(b\) \(\{(\text{role, leader}) , (\text{position, (10, 20)})\}\) and a message \(m\) \(\{(\text{subject, alert}), (\text{position}, (15, 20))\}\).

**Filter**: The filter identifies the entities according to their description \((e_d)\) and realizes the interaction between the concrete objects \((e_r)\). A filter \(f_j \in \mathcal{F}\) is a tuple \(f_j = (f_a, f_m, [f_c], n_f)\) with \(n_f\) filter name. The assertion \(f_a : \mathcal{A} \rightarrow \{T,F\}\) identifies the receiving agents (which agent), the assertion \(f_m : MSG \rightarrow \{T,F\}\) identifies the concerned messages (which message), and \(f_c : \mathcal{P}(\Omega) \rightarrow \{T,F\}\) is an optional set of assertions identifying other entities of the context (which context).

Each agent \(?r\) ("?" preceedings a letter denotes a variable) whose description validates \(f_a\) receives in its mailbox the message \(?m\) that satisfies \(f_m\) if there exists a set of entities in the \(?c\) such that \(f_c\) is true.

A filter is therefore valid for any tuple \((?r \in \mathcal{A}, ?m \in MSG[], ?c \subset \Omega)\)

For instance, in the crisis management application, the filter \(Fe\) sets the routing of the communication as follows ("=" is the comparison operator):

\[
\begin{align*}
- id &: \Omega \rightarrow N, \\
- role &: \Omega \rightarrow \{\text{coordinator, leader}\}, \\
- position &: \Omega \rightarrow N \times N, \\
- subject &: \Omega \rightarrow \{\text{alert, demand}\}, \\
- sender &: \Omega \rightarrow N.
\end{align*}
\]
The agents with the role \textit{leader} that are situated in the crisis origin:

\( f_a : [\text{role}(?r) = \text{leader}_D] \land [\text{position}(?r) = (0,0)] \)

should receive the alert messages:

\( f_m : [\text{subject}(?m) = \text{alert}] \land [\text{sender}(?m) = ?ide] \)

of the agent playing the role \textit{coordinator}:

\( f_C : [\text{id}(?e) = ?ide] \land [\text{role}(?e) = \text{coordinator}] \)

In this example, the description of the message sender (?e) that is identified thanks to the property \textit{sender} in the message belongs to the context. Agents wishing to send or receive a message, update their description in the communication platform and add/retract dynamically in/from the environment filters that involve them. Thus the environment supports simultaneously direct interaction (including dyadic, broadcast multi-cast and group communication) and indirect interaction (including mutual-awareness and overhearing). If the filter is added by the future receiver of the message then the interaction is indirect: the depositary agent defines which message it wants to receive. If the filter is added by the future message sender then the interaction is direct: the depositary agent defines which agent it wants to contact.

According to the state of the different descriptions within the environment, the triggered filter will enable the routing of the messages in the different interaction modes towards the corresponding targets.

Even if EASI offers an advanced communication management by identifying precisely the interaction context, it cannot be used by the agents in order to reason on the causes of the interaction. For instance, the filter \( F_e \) will permit the routing of messages but the reasons of this requirement is not expressed within EASI. For the filter \( F_e \), the choice of the communication mode may depend on the relations between the roles \textit{coordinator} and \textit{leader}:

- \textit{coordinator} sends messages to \textit{leaders} (direct mode) for dedicated messages whereas the \textit{leaders} listen to all the messages issued from the \textit{coordinator} (indirect mode). Using this knowledge, an agent could reason on the current interactions. For instance, the coordinator may choose a direct interaction to handle certain informations and indirect interaction for others. The leaders can deduce the importance to the informations according to the filter used to receive informations. The specification of communications within an organizational model would help the agents to relate communication filters to the reasons that cause the use of such a communication channel.

### 3 Extending \textsc{Moise} for Easi

In order to clearly specify the interaction modes used in EASI, we are going to enrich and extend the organization modeling language of \textsc{Moise} with a new dimension. This new dimension is called \textit{communication mode specification} (noted \( CS \)). It is dedicated to expressing the communication modes that will be used within the organization. As the other \textsc{Moise} dimensions, we keep it independent of \( SS \) and \( FS \). We use the same principle to connect it to the other dimensions and enrich the normative specification accordingly. The aim is to be able to
connect the communication modes to the structure and functioning of the organization by the way of norms. Those norms will be made accessible to the agents when interacting with other agents of the organization.

The organization specification is thus enriched into the following 4-uple: \( \langle SS, FS, CS, NS \rangle \) with \( CS \) communication modes specification and \( NS \) the modified normative specification. We detail these two components in what follows.

### 3.1 Communication modes specification

The specification \( CS \) is composed of the set of communication modes \( cm \in CS \) considered in the organization.

A communication mode is defined as: \( \langle \text{type, direction, protocol} \rangle \) with \( \text{type} \), the type of the communication mode (direct or indirect), direction, the message transmission direction (unidirectional or bidirectional), protocol, the interaction protocol that is used. The values of this last variable correspond to the names of the different interaction protocols that the designer wishes to be used and deployed in the organization (e.g. FIPARequest, PublishSubscribe, ...).

As we will see below, a communication mode qualifies the communication link defined in the structural specification between roles. The communication link is directed from the \textit{initiator} of the communication - source of the link - to the \textit{participant} - target of the link -. Therefore, a link can be considered as:

- a unidirectional channel, letting circulate messages in only one direction,
- a bidirectional channel, letting circulate messages in both directions from the initiator to the participant and inversely.

Orthogonal to these two directions, we consider the direct and indirect interaction models proposed within EASI.

In the crisis management application, we define, for instance, the two following communication modes \( cm_{d,b} \) and \( cm_{i,u} \):

\[
\begin{align*}
cm_{d,b}: & \langle \text{direct, bidirectional, FIPARequest} \rangle \\
cm_{i,u}: & \langle \text{indirect, unidirectional, PublishSubscribe} \rangle
\end{align*}
\]

where \( cm_{d,b} \) is used to directly ask for information whereas \( cm_{i,u} \) is used to provide information to agents that will consult it when they want.

### 3.2 Communication Norms

In order to bind communication link and communication mode as defined in \( CS \) by making explicit the deontic modalities attached to their use, we generalize the writing of the norms described in the MOISE initial version: \( \langle id, c, \rho, dm, object \rangle \) where \( id \) is norm identifier, \( c \) a the activation condition, \( \rho \) the role on which the deontic modality bears, \( dm \) the deontic modality (obligation or permission), \textit{object} the subject of the norm.
Object of a norm: The object of a norm \textit{object} is defined as the two following expressions:

- \textit{do}(m) in the case where a mission \textit{m} has to be executed - case initially considered in \textit{Moise},
- \textit{use}(l, cm, \alpha) in the case where the communication mode \textit{cm} should be used for the link \textit{l} in the context \textit{\alpha}.

Context: The context \textit{\alpha} defines the constraints bearing on the descriptions \epsilon_d of the entities \omega_i \in \Omega (cf. Sec. 2.2) involved in the interaction using this communication link: sender, receiver, message. It is also possible to add additional descriptions issued from other entities of the MAS (e.g. requirements of the agent, ...). It is thus possible to use a mission as context of use of the link or a particular goal as context of use of this link. We will define in the following section the way we express these constraints when describing how Easi has been specialized to handle \textit{Moise}. When \textit{\alpha}'s status is \textit{T} (true), the link is usable in any situation.

Let’s consider the communication link \textit{l}_1 used by the agents playing the role \textit{coordinator} towards agents playing the role \textit{leader} (eg. Fig. 1) in the crisis management application. Given the normative specification that we have defined, it is possible to bind to it the communication mode \textit{cm_{i,u}} defined above, by writing the following norm: \textit{n}_1 \langle \textit{n}_1, \textit{c}_1, \textit{coordinator}, \textit{obligation}, \textit{use}(\textit{l}_1, \textit{cm_{i,u}}, \textit{T}) \rangle with \textit{c}_1 : \textit{committed}(\textit{m}_1) to express that \textit{l}_1 ought to be used by agents playing the role \textit{coordinator} when they are committed to the mission \textit{m}_1. No particular context is attached to the use of the communication mode \textit{cm_{i,u}}.

We can also attach to this link another communication mode \textit{cm_{d,b}}, by defining a new norm \textit{n}_2 : \langle \textit{n}_2, \textit{c}_2, \textit{coordinator}, \textit{obligation}, \textit{use}(\textit{l}_1, \textit{cm_{d,b}}, \textit{\alpha}_2) \rangle with \textit{c}_2 : \textit{committed}(\textit{m}_4) by specifying a context \textit{\alpha}_2 (cf. following section for the syntax) stating that the communication on the link \textit{l}_1 takes place for the sending of messages to agents belonging to group \textit{CIGT} (Center of the Engineering and Management of the Traffic). The link \textit{l}_1 can also be bound to the same constraints but for communication in the context \textit{\alpha}_3 stating the sending of messages from the agent playing role \textit{coordinator} to agent belonging to group \textit{TNM}: \langle \textit{n}_3, \textit{c}_2, \textit{coordinator}, \textit{obligation}, \textit{use}(\textit{l}_1, \textit{mc_{d,b}}, \textit{\alpha}_3) \rangle.

In the following, we will need to access to the different features of a communication link from the structural specification. We will use pointed notation \textit{l}_j.initiator (resp. \textit{l}_j.participant) to access to the source role (resp. target) of the link \textit{l}_j, and \textit{l}_j.group to access to the group in which \textit{l}_j is defined.

4 Specializing Easi for \textit{Moise}

Our objective is to generate filters for the communication environment from the specifications defined in the organization modeling language. These filters use informations on the organization. These informations should be stored in the description of the entities that are managed by the communication environment.
in order to be accessible to the filters. In this section, we identify and define the necessary properties for describing the agents and messages in the communication environment. Then we describe a generic filter generated from the communication norms that we just defined in the previous section.

4.1 Properties

In order to connect organization and interaction, it is necessary to give a minimal description of an agent, of a message while incorporating this new dimension in them. Given the definition of an entity in section 2.2, we define the following properties that are accessible in the environment for each type of entity.

Agent Properties: The description of an agent is at least composed of the id and org properties, where:

- id returns the identifier of the agent \(id : A \rightarrow ID_A\) with \(ID_A\) set of agent identifiers,
- org returning the subset of organizational descriptions coming from the participation of the agent to the organization \(org : A \rightarrow \mathcal{P}(OC)\) with \(OC\) set of organization descriptions).

An organization description \(oc_i \in OC\) is defined by: \(oc_i = \langle ig : g, r, m, go \rangle\) with \(ig \in IG, g \in \{rg\} \cup rg.subgroups, r \in R, m \in M, go \in G\). \(ig\) is a concrete group created from the group specification \(g\) defined in the SS of the organization. The parameter \(rg\) and the sets \(R, M, G\) are defined in the organization specification (cf. Sec. 2.1).

For instance, in the crisis management application, the agent \(a\) described by \(\text{org}(a) = \{\langle g_1 : \text{Decision-making}, \text{leader}_D, m_2, b_2 \rangle, \langle g_2 : \text{DDE}, \text{leader}_S, m_1, b_1 \rangle\}\), belongs to the group \(g_1\) of type Decision-making and to a group \(g_2\) of type DDE, in which it plays respectively role \(\text{leader}_D\), committed to mission \(m_2\), trying to achieve goal \(b_2\) and the role \(\text{leader}_S\), committed to the mission \(m_1\) trying to achieve the goal \(b_1\).

This description of an agent is minimal. We defined two management modes. Being related to the organization, these properties can be managed without being intrusive: management by the organization management infrastructure. However, if this set is complemented by specific properties related to the internal state of the agents, their management is ensured by the agents themselves. For instance, a property availability returning the availability of the agent has a value that is related to the choice of the agent itself.

Message Properties: In the same way, we specialize the description of a message with the following set of minimal properties sender, receiver, subject, rc, sc where:

- \(\text{sender} : MSG \rightarrow ID_A\),
- \(\text{receiver} : MSG \rightarrow \mathcal{P}(ID_A) \cup \{\text{unknown}\}\),
- \(\text{subject} : MSG \rightarrow D_{subject} \cup \{\text{unknown}\}\), with \(D_{subject} = G \cup R \cup \{\text{expression}\}\),
  expression is a string,
rc: $MSG \rightarrow OC \cup \{unknown\}$ being the reception context,
sc: $MSG \rightarrow OC \cup \{unknown\}$ being the sending context.

Using these properties, the sender gives informations on the organizational context in which the interaction takes place. For a message, each of these properties can receive a value or the value $unknown$. The more the sender specifies values of properties, the more precise will be the filter that can be used for the routing. We impose that the property $sender$ doesn’t get a value $unknown$ in order to avoid anonymous messages.

Given these different properties, we have now the possibility of a routing ranging from indirect interaction, based on only the identifier of the sender, to one, focused on a subset of receivers ($receiver$) in a particular organizational context ($rc$), with a sender being in a given organizational context ($sc$) and the message with a particular content ($subject$).

The sender can also decide to define patterns for conditioning the routing along different organizational contexts. To this aim, it can use the symbol ‘?’ as value for an element of the organizational context. This symbol denotes that the value is not a constraint in the choice of the tuples.

For instance, in the crisis management application, the expression $\langle \circlearrowleft DDE, \circlearrowleft m_2, \circlearrowleft \rangle$ defines an organizational pattern of $OC$ such that the concrete group must be of type $DDE$ and the mission is $m_2$ whatever are the values for roles and goals. The message $mes_1$ described below means that the sender whose identifier is $a_1$ and having the goal $b_2$ (sending context) sends a message to the agents $a_2$ and $a_4$. In this case, the processing of the message is not constrained by the organizational states of the participating agents. They only have to be trying to achieve the organizational goal $b_2$. $\langle (sender, a_1), (receiver, \{a_2, a_4\}), (subject, demand), (rc, \langle \circlearrowleft, \circlearrowleft, b_2 \rangle), (sc, \langle \circlearrowleft, \circlearrowleft, b_2 \rangle) \rangle$

For the sender, these are only possibilities since the routing of the message depends on the filters that are installed in the communication environment.

In fact, according to the filters that are installed in it, the routing of the message can lead to different situations: interaction as intended by the sender, no interaction or interaction not intended by the sender. For instance, the agent $a_2$ can receive the message although it doesn’t have the goal $b_2$ in the case there exists a filter enabling the reception of messages from the agent $a_1$, whatever are the values for the properties of the message.

In each message is stored the organizational context of its sending in order to enable the agents to filter them. An agent can thus choose to receive messages or to route them according to their organizational contexts without being imposed their use. Moreover, this definition of messages enables to consider the evolution of the organization state. Thus, a message kept in the environment can still be received by an agent in case of change of the organization state. For instance, an agent can be interested by any message whose receiving context concerns a role that it just endorsed. It is useful to keep an history of the past interaction
to better understand the current situation. Another advantage is to avoid that a message is missed because it has been sent before the agent has endorsed the role. In order to avoid a risk of confusion between messages, a property related to the time value of there emission or related to there life time can be added to the message description. This choice belongs to the system designer and is out of the scope of this paper.

4.2 From Communication Norms to Environment Filters

The activation of a norm for a communication link leads to the generation and addition of a filter in the environment. This filter is called normative filter. It corresponds to the exact translation of the norm as it is instantiated by the organization management system. Thanks to the organization management system, the agents are aware of the norm activation. Besides to the normative filters, the communication environment contains also filters set by the agents according to their activity in the system. In case of direct interaction, the sender knows that it can reach the agents identified as receiver in the norm. In case of indirect interaction, the receiver knows that it can receive messages identified in the norm.

A normative filter uses all the possible informations coming from the organizational specification and routes a message according to its \( sc \) and \( rc \) properties. The property \( receiver \) is not used in the generation process of a normative filter since it requires that the sender knows the identifiers of the agents. This is a too strong hypothesis. The same way, since the routing comes from the activation of a norm, the filter cannot constrain the subject of the message (\( subject \)) except additional conditions in the norms (context \( \alpha \) of the object of the norm). The filter identifies a state of the context corresponding to the interaction. It is identical in the direct and indirect cases. We then propose a generating pattern that will be specialized for each activated communication norm.

Access to the organizational specification: The normative filter is created when the norm is activated as follows.

Let’s first define the functions \( initiator \) and \( participant \) that access to the agents involved in the communication link defined in the object of the norm.

\[
\text{initiator} : \mathcal{O} \rightarrow A \quad \text{participant} : \mathcal{O} \rightarrow A
\]

These functions return, for an instantiated norm, the agent (\( \text{initiator} \)), who initiates, or the one (\( \text{participant} \)), who participates, to the interaction. From these two functions, we express constraints on the descriptions of the agents. For instance \( \text{arg}(\text{initiator}(n_j)) \) makes possible to access to the organizational context attached to the description of the agent initiating the communication in the context of the instantiated norm \( n_j \) in which it is involved.

Let’s define the predicate \( \text{achieves}_\alpha \) that is automatically generated from the constraints expressed in the context \( \alpha \) of the object of a norm. This predicate checks that the context is satisfied given the initiator, participant, message and entity descriptions in the environment, given \( \alpha : \)

\[
\text{achieves}_\alpha : A \times MSG \times A \times \mathcal{P}(\Omega) \rightarrow \{T, F\}
\]
Given the previous definitions, we are able now to express the generic normative filter \( f_{n_k}(?p, ?m, \{?i, C\}) \) for the receiver \(?p\) of the message \(?m\) sent by \(?i\) in the context \(C\). This filter has been generated from the activation of the norm \(n_k\). The object of the norm bears on the communication link \(l_j\). It is composed of assertions \(f_a\) that identifies the receiver of the message \(?p\) according to its organizational context, \(f_m\) that identifies the message \(?m\) according to its organizational context and \(f_c\) that identifies the organizational context of the sender and the constraints \(\alpha\) of the norm \(n_k\).

\[
\begin{align*}
  f_a : \langle \text{org}(?p) \ni \langle ?x : l_j.\text{group}, l_j.\text{participant}, \ldots \rangle \rangle \\
  f_m : \langle \text{sender}(?m) = \text{id}(?i) \rangle \land \langle \text{sc}(?m) = \langle ?y : l_j.\text{group}, l_j.\text{initiator}, \ldots \rangle \rangle \land \langle \text{rc}(?m) = \langle ?x : l_j.\text{group}, l_j.\text{participant}, \ldots \rangle \rangle \\
  f_c : \langle \text{org}(?i) \ni \langle ?y : l_j.\text{group}, l_j.\text{initiator}, \ldots \rangle \rangle \land \langle \text{achieves}_{\alpha}(?p, ?m, ?i, C) \rangle
\end{align*}
\]

Let's consider again the norm \(n_2\) of the crisis management application: \( \langle n_2, \text{committed}(m_4), \text{coordinator}, \text{obligation}, \text{use}(l_1, cm_{d,b}, a_2) \rangle \) with \(a_2: \{\mathcal{.}: \text{CIGT}, \ldots \} \in \text{org}(\text{participant}(n_2))\). The interaction is a direct and bidirectional one (cf. \(cm_{d,b}\) of \(n_2\)). The sending agent deposits the first message. The two necessary filters have been generated and added thanks to the activation of \(n_2\).

The normative filter generated for \(n_2\) for the interaction from initiator to participant is \(f_{n_2}(?p, ?m, \{?i, C\})\) : where:

\[
\begin{align*}
  f_a : \langle \text{org}(?p) \ni \langle ?x : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle \\
  f_m : \langle \text{sender}(?m) = \text{id}(?i) \rangle \land \langle \text{sc}(?m) = \langle ?y : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle \land \langle \text{rc}(?m) = \langle ?x : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle \\
  f_c : \langle \text{org}(?i) \ni \langle ?y : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle \land \langle \text{org}(?p) \ni \langle \mathcal{.}: \text{CIGT}, \ldots \rangle \rangle
\end{align*}
\]

The normative filter from the participant to the initiator is \(f_{n_2}(?i, ?m, \{?p, C\})\) : \(^7\)

\[
\begin{align*}
  f_a : \langle \text{org}(?i) \ni \langle ?x : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle \\
  f_m : \langle \text{sender}(?m) = \text{id}(?p) \rangle \land \langle \text{sc}(?m) = \langle ?y : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle \land \langle \text{rc}(?m) = \langle ?x : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle \\
  f_c : \langle \text{org}(?i) \ni \langle ?y : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle \land \langle \text{org}(?p) \ni \langle \mathcal{.}: \text{CIGT}, \ldots \rangle \rangle
\end{align*}
\]

This way, for two agents participating to the same concrete group, the message sent by the initiator agent \(a_1\) processed by the filter \(f_{n_2}\) will have the following description: \(\langle \text{sender}, \text{id}(a_1) \rangle, \langle \text{rc}, \langle g_1 : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle, \langle \text{sc}, \langle g_1 : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle\)

The message sent by the participant agent \(a_2\), processed by \(f_{n_2}\) will have the following description: \(\langle \text{sender}, \text{id}(a_2) \rangle, \langle \text{rc}, \langle g_1 : \text{Decision-making}, \text{leader}_D, \ldots \rangle \rangle, \langle \text{sc}, \langle g_1 : \text{Decision-making}, \text{coordinator}, \ldots \rangle \rangle\)

With these two filters, a communication channel has been created between agents having the roles coordinator and responsible in the group \(\text{CIGT}\). The interaction model \text{EASI} has make possible to elaborate these filters. The \text{MOISE} model has made possible its use.

\(^7\) we continue to use the variable \(?p\) for the participant in the interaction and \(?i\) for the initiator given that the agent which is identified by the variable \(?i\) who receives the message sent by \(?p\).
5 Example

In this section, we illustrate and discuss the expressing capabilities of our proposal going back to the interaction modes attached to the communication link \( l_1 \) issued of the communication norms \( n_1, n_2, n_3 \) in the crisis management application described in the paper.

\[ \langle n_1, c_1, \text{coordinator}, \text{obligation}, \text{use}(l_1, \text{cm}_{i,u}, T) \rangle \text{ with } c_1 : \text{committed}(m_1) \]

\[ \langle n_2, c_2, \text{coordinator}, \text{obligation}, \text{use}(l_1, \text{cm}_{d,b}, \alpha_2) \rangle \text{ with } c_2 : \text{committed}(m_4) \]

\[ \alpha_2 = (\text{CIGT}, \ldots, \ldots) \in \text{org}(\text{participant}(n_2)) \]

\[ \langle n_3, c_3, \text{coordinator}, \text{obligation}, \text{use}(l_1, \text{cm}_{d,b}, \alpha_3) \rangle \text{ with } c_3 : \text{committed}(m_4) \]

\[ \alpha_3 = (\text{TNM}, \ldots, \ldots) \in \text{org}(\text{participant}(n_3)) \]

On these three norms, the differences bear on the activation conditions of the norm \( c_x \), the communication mode \( \text{cm}_{x,y} \) and the communication context specified in the object.

The norm \( n_1 \) whose activation condition bears on the management of the crisis (mission \( m_1 \)) is activated during all the crisis management. The norms \( n_2 \) and \( n_3 \) are not active since the agents on which the norms bear are committed on the mission \( m_4 \).

The predicate achieves\(_x\) of the normative filter \( f_{n_1} \) generated from the norm \( n_1 \) is always true (\( \alpha_1 = T \)). According to this norm, all the agents playing the role leader\(_D\) (target of link \( l_1 \)) must consult the informations set available by any agent playing the role coordinator. The norm \( n_2 \) imposes a direct interaction in the context of mission \( m_4 \) so that the coordinator is able to get informations on the state of the transportation network. According to this norm, any agent playing the role coordinator can reach any agent playing the role leader\(_D\) (target of link \( l_1 \)) and being a member of concrete group of type CIGT. The normative filter \( f_{n_2} \) described in the previous section expresses these constraints. For the same mission \( m_4 \), the coordinator requires information on the available resources in the services TNM. The normative filter \( f_{n_3} \) resulting from the activation of norm \( n_3 \), enables the coordinator to reach any leader of each traffic network management service (TNM).

In our example, if the missions \( m_1, m_4 \) are under examination, the normative filters corresponding to the three norms are simultaneously present in the environment. From the point of view of the agent playing the role coordinator, it means that it can route messages directly to the agents who are leader\(_D\) in groups of type CIGT (\( n_2 \)) and broaden their demand to agents playing the role leader\(_D\) in the groups of type TNM (\( n_3 \)) given its needs.

Let’s turn to the agents playing the role leader\(_S\) in the group Decision-making. If involved in the role leader\(_S\) within the groups CIGT and TNM (let’s notice that this situation is possible thanks to the compatibility link between both roles), the agents will receive the requests from the agent playing the role coordinator and will be able to know that this is a direct interaction issued from the coordinator. The agents will be able to answer to this agent by using the normative filter created in case of bidirectional interaction. Thanks to norm \( n_1 \), every agent playing the role leader\(_D\) will receive the messages sent by the agents playing
the role coordinator via the filter $f_{n_1}$, building a common and shared knowledge (indirect interaction). According to their processing activity, the agents will be more or less aware of these messages.

This short example that we can’t detail more, shows the richness of expressiveness of the interaction modes made possible by combining EASI and MOISE as described in this paper.

6 Related work

To our knowledge, there doesn’t exist a similar support to interaction enabling, for the same communication, to consider simultaneously the direct and indirect interaction modes.

Considering related works to the indirect interaction, the general principle consists in the use of a shared data space that is integrated or not to the environment [8]. In this approach, the tuples that are deposit by the sender in the shared space are compared to patterns expressing the needs of the receivers. These works are focused on the accompanying coordination language and don’t consider, at any moment, the organization or the state of the agents.

Dealing with the direct interaction model, several works propose to use an organizational structure in order to manage the communications. In [1], the agents are organised in a hierarchy where each level knows the skills of the agents belonging to the lower level in order to make possible for the sender, a routing of the messages according to the skills. However, it is not an organizational model that is usable by the agents. In the AGR model [3], the organization constrains the interactions according to the groups to which the agents participate. It supports a routing of the message according to the organizational model (group, role). However, the only interaction mode is the direct one and the agents don’t have access to an explicit description of the different specifications.

Normative organization models have been proposed in the literature in order to regulate and control the communication between agents. However the specifications address the interaction protocols, i.e. the coordination of the interaction instead of interaction modes. The only considered interaction mode is the direct one (e.g. ISLANDER [2]). They don’t consider the interaction at the level addressed in this paper.

7 Conclusions

In this paper, we have proposed a specification of interaction modes between agents within an organization. For that aim, we have extended and enriched the organization modeling language of the MOISE framework. We have also shown how the specifications have been used to generate and to configure dynamically the communication environment supported by the EASI platform. We have illustrated the use of this proposal in a crisis management application.

In the future, we intend to extend the considered interaction modes to overhearing. We will also consider the communication between groups by extending
the scope of communication to groups by enriching and modifying the structural specification of Moise. Thanks to these new primitives in the organization specification, we can turn to the development of reasoning mechanisms at the agent level to make agents able to reason on the interaction modes that they can use within the organization.

Acknowledgement

We would like to thank D. Trabelsi, H. Hadioui and J.F. Hübner for the fruitful discussions about the content of this paper.

References

Norm Refinement and Design through Inductive Learning

Domenico Corapi¹, Marina De Vos², Julian Padget², Alessandra Russo¹, and Ken Satoh³

¹ Department of Computing, Imperial College London
{d.corapi,a.russo}@imperial.ac.uk
² Department of Computer Science, University of Bath
{mdv,jap}@cs.bath.ac.uk
³ National Institute of Informatics
ksato@nii.ac.jp

Abstract. In the physical world, the rules governing behaviour are debugged by observing an outcome that was not intended and the addition of new constraints intended to prevent the attainment of that outcome. We propose a similar approach to support the incremental development of normative frameworks (also called institutions) and demonstrate how this works through the validation and synthesis of normative rules using model generation and inductive learning. This is achieved by the designer providing a set of use cases, comprising collections of event traces that describe how the system is used along with the desired outcome with respect to the normative framework. The model generator encodes the description of the current behaviour of the system. The current specification and the traces for which current behaviour and expected behaviour do not match are given to the learning framework to propose new rules that revise the existing norm set in order to inhibit the unwanted behaviour. The elaboration of a normative system can then be viewed as a semi-automatic, iterative process for the detection of incompleteness or incorrectness of the existing normative rules, with respect to desired properties, and the construction of potential additional rules for the normative system.

1 Introduction

Norms and regulations play an important role in the governance of human society. Social rules such as laws, conventions and contracts prescribe and regulate our behaviour, however it is possible for us to break these rules at our discretion and face the consequences. By providing the means to describe and reason about norms in a computational context, normative frameworks (also called institutions or virtual organisations) may be applied to software systems allowing for automated reasoning about the consequences of socially acceptable and unacceptable behaviour, by monitoring the permissions, empowerment and obligations of the participants and generating violations when norms are not followed.

* This work is partially supported through the EU Framework 7 project ALIVE (FP7-IST-215890), and the EPSRC PRiMMA project (EP/F023294/1).
The formal model put forward in [9] and its corresponding operationalisation through Answer Set Programming (ASP) [3, 18] aims to support the top-down design of normative frameworks. AnsProlog is a knowledge representation language that allows the programmer to describe a problem and required properties on the solutions in an intuitive way. Programs consist of rules interpreted under the answer set semantics. Answer set solvers, like CLASP[17] or SMODELS[25], can be used to reason about the given AnsProlog specification, by returning acceptable solutions in the form of traces, as answer sets. In a similar way, the correctness of the specification with respect to given properties can be verified.

Currently, the elaboration of behavioural rules and norms is an error-prone process that relies on the manual efforts of the designer and would, therefore, benefit from automated support. In this paper, we present an inductive logic programming (ILP) [24] approach for the extraction of norms and behaviour rules from a set of use cases. The approach is intended as a design support tool for normative frameworks. Complex systems are hard to model and even if testing of properties is possible, sometimes it is hard to identify missing or incorrect rules. In some cases, e.g. legal reasoning, the abstract specification of the system can be in part given in terms of specific instances and use cases that ultimately drive the design process and are used to assess it. We propose a design support tool that employs use-cases, i.e. traces together with their expected normative behaviour, to assist in the revision of a normative framework. The system is correct when none of the traces are considered dysfunctional, i.e. they match the expected normative behaviour. When a dysfunctional trace is encountered the normative specification needs to be adjusted: the task is to refine the given description by learning missing norms and/or behavioural rules that, added to the description, entail the expected behaviour over the traces. We show how this task can be naturally represented as a non-monotonic ILP problem in which the partial description of the normative system provides the background knowledge and the expected behaviour comprises the examples. In particular, we show how a given AnsProlog program and traces can be reformulated into an ILP representation that makes essential use of negation in inducing missing parts of the specification. As the resulting learning problem is inherently non-monotonic, we use a non-monotonic ILP system, called TAL [12], to compute the missing specification from the traces and the initial description.

Given the declarative nature of ASP, the computational paradigm used for our normative frameworks, we needed to adopt a declarative learning approach as we aim to learn declarative specifications. This differs from other approaches, such as reinforcement learning whereby norms or policies are learned as outcomes of estimation and optimisation processes. Such types of policies are not directly representable in a declarative format and are quite different in nature from the work reported here.

The paper is organised as follows. Section 2 presents some background material on the normative framework, while Section 3 introduces the non-monotonic ILP system used in our proposed approach. Section 4 describes the AnsProlog modelling of normative frameworks. Section 5 illustrates how the revision task can be formulated into an ILP problem, and how the generated ILP hypothesis can be reformulated as norms and behaviour rules within the AnsProlog representation. In Section 6 we illustrate the flexibility and expressiveness of our approach through a number of different par-
tial specifications of a reciprocal file sharing normative framework. Section 7 relates our approach to existing work on learning norms with respects to changing/improved requirements. We conclude with a summary and remarks about future work.

2 Normative Frameworks

The concept of normative framework has become firmly embedded in the agent community as a necessary foil to the essential autonomy of agents, in just the same way as societal conventions and legal frameworks have grown up to constrain people. In both the physical and the virtual worlds, and the emerging combination of the two, the arguments in favour centre on the minimisation of disruptive behaviour and supporting the achievement of the goals for which the normative framework has been conceived and thus also the motivation for submission to its governance by the participants. While the concept remains attractive, its realisation in a computational setting remains a subject for research, with a wide range of existing logics [29, 1, 7, 9, 32] and tools [26, 14, 19].

2.1 Formal Model

To provide context for this paper, we give an outline of a formal event-based model for the specification of normative frameworks that captures all the essential properties, namely empowerment, permission, obligation and violation. Extended presentations appear in [9] and [10].

The essential elements of our normative framework are: (i) events ($E$), that bring about changes in state, and (ii) fluents ($F$), that characterise the state at a given instant. The function of the framework is to define the interplay between these concepts over time, in order to capture the evolution of a particular institution through the interaction of its participants. We distinguish two kinds of events: normative events ($E_{norm}$), that are the events defined by the framework and exogenous ($E_{ex}$), that are outside its scope, but whose occurrence triggers normative events in a direct reflection of the “counts-as” principle [21]. We further partition normative events into normative actions ($E_{act}$) that denote changes in normative state and violation events ($E_{viol}$), that signal the occurrence of violations. Violations may arise either from explicit generation, from the occurrence of a non-permitted event, or from the failure to fulfil an obligation. We also distinguish two kinds of fluents: normative fluents that denote normative properties of the state such as permissions $P$, powers $W$ and obligations $O$, and domain fluents $D$ that correspond to properties specific to the normative framework itself. The set of all fluents is denoted as $F$. A normative state is represented by the fluents that hold true in this state. Fluents that are not presented are considered to be false. Conditions on a state are therefore expressed by a set of fluents that should be true or false. The set of possible conditions is referred to as $X = 2^F \cup \neg F$.

Changes in state are achieved through the definition of two relations: (i) the generation relation, which implements counts-as by specifying how the occurrence of one (exogenous or normative) event generates another (normative) event, subject to the empowerment of the actor and the conditions on the state, and (ii) the consequence relation. This latter specifies the initiation and termination of fluents subject to the performance of some action in a state matching some expression. The generation relation is formally
defined as $G : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{E}_{\text{norm}}}$, and the consequence relation as $C : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}}$. The fluents to be initiated as a result of an event $E$ are often denoted by $C^\uparrow(\phi, E)$ while the ones to be terminated are denoted by $C^\downarrow(\phi, E)$.

The semantics of our normative framework is defined over a sequence, called a trace, of exogenous events. Starting from the initial state, each exogenous event is responsible for a state change, through initiation and termination of fluents. This is achieved by a three-step process: (i) the transitive closure of $G$ with respect to a given exogenous event determines all the generated (normative) events, (ii) to this all violations of events not permitted and obligations not fulfilled are added, giving the set of all events whose consequences determine the new state, (iii) the application of $C$ to this set of events identifies all fluents that are initiated and terminated with respect to the current state so giving the next state. For each trace, we can therefore compute a sequence of states that constitutes the model of the normative framework for that trace. This process is realised as a computational model through Answer Set Programming (see Section 4) and it is this representation that is the subject of the learning process described in Section 5.

3 Learning

Inductive Logic Programming (ILP) [24] is a machine learning technique concerned with the induction of logic theories from (positive and negative) examples and has been successfully applied to a wide range of problems [15]. Automatic induction of hypotheses represented as logic programs is one of the distinctive features of ILP. Moreover, the use of logic programming as representation language allows a principled representation of background information relevant to the learning. To refine normative theories we employ an ILP learning system, called TAL [12], that is able to learn non-monotonic theories, and can be employed to perform learning of new rules and the revision of existing rules. The TAL approach is based on mapping a given inductive problem into an abductive reasoning process. The current implementation of TAL relies on an extension of the abductive procedure SLDNFA [13] and preserves its semantics.

Definition 1. A non-monotonic ILP task is defined as $\langle E, B, S \rangle$ where $E$ is a set of ground positive or negative literals, called examples, $B$ is a background normal theory and $S$ is a set of clauses called language bias. The normal theory $H \in \text{ILP}(E, B, S)$, called hypothesis, is an inductive solution for the task $\langle E, B, S \rangle$, if $H \subseteq S$, $H$ is consistent with $B$ and $B \cup H \models E$.

$B$ and $H$ are normal theories and thus support negation as failure. The choice of an appropriate language bias is critical. In TAL the language bias $S$ is specified by means of mode declarations [?].

Definition 2. A mode declaration is either a head or body declaration, respectively $\text{modeh}(s)$ and $\text{modeb}(s)$ where $s$ is called a scheme. A scheme $s$ is a ground literal containing place-markers. A place-marker is a ground function whose functor is one of the three symbols ‘+’ (input), ‘-’ (output), ‘#’ (constant) and the argument is a constant called type.
Given a schema $s$, $s^*$ is the literal obtained from $s$ by replacing all place-markers with different variables $X_1, ..., X_n$. A rule $r$ is compatible with a set $M$ of mode declarations iff (a) there is a mapping from each head/body literal $l$ in $r$ to a head/body declaration $m \in M$ with schema $s$ such that each literal is subsumed by its corresponding $s^*$; (b) each output place-marker is bound to an output variable; (c) each input place-marker is bound to an output variable appearing in the body or to a variable in the head; (d) every constant place-marker is bound to a constant; (e) all variables and constants are of the corresponding type. From a user perspective, mode declarations establish how rules in the final hypotheses are structured, defining literals that can be used in the head and in the body of a well-formed hypothesis. Although we show $M$ in the running example of this paper for reference, the mode declarations can be concealed from the user and derived automatically. They can be optionally refined to constrain the search whenever the designer wants to employ useful information on the outcome of the learning to reduce the number of alternative hypotheses or improve performance.

4 Modelling Normative Frameworks

While the formal model of a normative framework allows for clear specification of a normative system, it is of little support to designers or users of these systems. In order to be able to do so, computational tools are needed. The first step is a computational model equivalent to the formal model. We have opted for a form of logic programming, called Answer Set Programming (ASP)[18]. Here we only present a short flavour of the language AnsProlog, and the interested reader is referred to [3] for in-depth coverage.

AnsProlog is a knowledge representation language that allows the programmer to describe a problem and the requirements on the solutions in an intuitive way, rather than the algorithm to find the solutions to the problem. The basic components of the language are atoms, elements that can be assigned a truth value. An atom can be negated using negation as failure so creating the literal not $a$. We say that not $a$ is true if we cannot find evidence supporting the truth of $a$. If $a$ is true then not $a$ is false and vice versa. Atoms and literals are used to create rules of the general form: $a \leftarrow B, \text{not } C$, where $a$ is an atom and $B$ and $C$ are set of atoms. Intuitively, this means if all elements of $B$ are known/true and no element of $C$ is known/true, then $a$ must be known/true. We refer to $a$ as the head and $B \cup \text{not } C$ as the body of the rule. Rules with empty body are are called facts; A program in AnsProlog is a finite set of rules.

The semantics of AnsProlog are defined in terms of answer sets, i.e. assignments of true and false to all atoms in the program that satisfy the rules in a minimal and consistent fashion. A program has zero or more answer sets, each corresponding to a solution.

4.1 Mapping the formal model into AnsProlog

In this section we only provide a summary description of how the formal institutional model is translated in to AnsProlog. A full description of the model can be found in [9] together with completeness and correctness of model with respect to traces. Each program models the semantics of the normative framework over a sequence of $n$ time
instants such that $t_i: 0 \leq i \leq n$. Events are considered to occur between these snapshots, where for simplicity we do not define the intervals at which events occur explicitly, and instead refer to the time instant at the start of the interval at which an event is considered to occur. Fluents may be true or false at any given instant of time, so we use atoms of the form $\text{holdsat}(f, t_i)$ to indicate that fluent $f$ holds at time instant $t_i$. In order to represent changes in the state of fluents over time, we use atoms of the form $\text{initiated}(f, t_i)$ and $\text{terminated}(f, t_i)$ to denote the fact that fluent $f$ was initiated or terminated, respectively, between time instants $i$ and $i + 1$. We use atoms of the form $\text{occurred}(e, t_i)$ to indicate that event $e \in E$ is considered to have occurred between instant $t_i$ and $t_{i+1}$. These atoms denote events that occur in an external context or are generated by the normative framework. For exogenous events we additionally use atoms of the form $\text{observed}(e, t_i)$ to denote the fact that $e$ has been observed.

The mapping of a normative framework consists of three parts: a base component which is independent of the framework being modelled, the time model and the framework specific component. The independent component deals with inertia of the fluents, the generation of violation events of un-permitted actions and unsatisfied obligations. The time model defines the predicates for time and is responsible for generating a single observed event at every time instance. In this paper we will focus solely on the representation of the specific features of the normative framework.

In order to translate rules in the normative framework relations $G$ and $C$, we must first define a translation for expressions which may appear in these rules. The valuation of a given expression taken from the set $\mathcal{X}$ depends on which fluents may be held to be true or false in the current state (at a give time instant). We translate expressions into ASP rule bodies as conjunctions of extended literals using negation as failure for negated expressions.

With all these atoms defined, mapping the generation function and the consequence relation of a specific normative framework becomes rather straightforward. The generation function specifies that an normative event $e$ occurs at a certain instance ($\text{occurred}(e, t)$) when another event $e'$ occurs, the event $e$ is empowered ($\text{holdsat}(\text{pow}(e), t)$ and a set of conditions on the state are satisfied ($\text{holdsat}(f, t)$ or $\text{not holdsat}(f, t)$). The rules for initiation ($\text{initiated}(f, t)$) and termination ($\text{terminated}(f, t)$) of a fluent $f$ are triggered when a certain event $e$ occurs ($\text{occurred}(e, t)$) and a set of conditions on the state are fulfilled. The initial state of our normative framework is encoded as simple facts ($\text{holdsat}(f, i00)$).

Figure 1 gives a summary of all $\text{AnsProlog}$ rules that are generated for a specific normative framework, including the definition of all the fluents and events as facts. For a given expression $\phi \in \mathcal{X}$, we use the term $EX(\phi, T)$ to denote the translation of $\phi$ into a set of ASP literals of the form $\text{holdsat}(f, T)$ or $\text{not holdsat}(f, T)$.

In situations where the normative system consists of a number of agents whose actions can be treated in the same way (e.g. the rules for borrowing a book are the same for every member of alibrary) or where the state consists of fluents that can be treated in a similar way (e.g. the status of book), we can parameterise the events and fluents. This is represented in the $\text{AnsProlog}$ program by function symbols (e.g $\text{borrowed}(\text{Agent}, \text{Book})$) rather than terms. To allow for grounding, extra atoms to
\[ p \in \mathcal{F} \iff \text{ifluent}(p) \]
\[ e \in \mathcal{E} \iff \text{event}(e) \]
\[ e \in \mathcal{E}_{\text{ex}} \iff \text{evtype}(e, \text{obs}) \]
\[ e \in \mathcal{E}_{\text{act}} \iff \text{evtype}(e, \text{act}) \]
\[ e \in \mathcal{E}_{\text{viol}} \iff \text{evtype}(e, \text{viol}) \]
\[ C^\uparrow(\phi,e) = P \iff \forall p \in P \cdot \text{initiated}(p,T) \leftrightarrow \text{occurred}(e,I), \text{EX}(\phi,T). \]
\[ C^\downarrow(\phi,e) = P \iff \forall p \in P \cdot \text{terminated}(p,T) \leftrightarrow \text{occurred}(e,I), \text{EX}(\phi,T). \]
\[ G(\phi,e) = E \iff g \in E, \text{occurred}(g,T) \leftrightarrow \text{occurred}(e,T), \text{holdsat}(\text{pow}(e),I), \text{EX}(\phi,T). \]
\[ p \in S_0 \iff \text{holdsat}(p,i00). \]

**Fig. 1.** The translation of normative framework specific rules into AnsProlog

Ground these variables need to be added. Grounded versions of the atoms also need to be added to the program. An example of this can be found in Section 6.

### 5 Learning Normative Rules

#### 5.1 Methodology

The development process is supported by a set of use cases \( U \). Use cases represent instances of executions that are known to the designer and that drive the elaboration of the normative system. If the current formalisation of the system does not match the intended behaviour in the use case then the formalisation is still not complete or incorrect. Each use case \( u \in U \) is a tuple \( \langle T,C,O \rangle \) where \( T \) is a trace that specifies all the exogenous events occurring at all the time points considered (\( \text{observed}(e,T) \)); \( C \) are ground \( \text{holdsat} \) or \( \text{occurred} \) facts that the designer believes to be important and represents the conditional expected output; \( O \) are ground \( \text{holdsat} \) and \( \text{occurred} \) literals that represent the expected output of the use case.

The design process is iterative. A current formalisation of the model in AnsProlog is tested against a set of use cases. Together with the AnsProlog specification of the normative framework we add the observed events and a constraint indication that no answer set that does not satisfy \( O \) is acceptable. The latter is done by adding a constraint containing the negation of all the elements in \( O \). If for some use cases the solver is not able to find an answer set (returns unsatisfiable), then a revision step is performed. All the use cases and the current formalisation are given as input to TAL. Possible revisions are provided to the designer who ultimately chooses which is the most appropriate. The success of the revision step depends on the state of the formalisation of the model. The set of supporting use cases can be extended as the design progresses to more accurate models.

In this paper we focus on the learning step and we show how a non-monotonic ILP system can be used to derive new rule. Refining existing rules (i.e. deleting rules or adding and delete conditions in rules) is a straightforward extension of the current framework. Though we do not discuss it in this paper, revision can be performed by extending the original rules with additional predicates that extend the search to deletion of conditions in rules and to exceptions as shown in [11].
5.2 Mapping ASP to ILP

The differences between the AnsProlog program and the translation into a suitable representation for TAL is procedural and only involves syntactic transformations. Thus the difference in the two representations only consists in how the inference is performed. The two semantics coincide since the same logic program is encoded and the mapping of a normative framework has exactly one answer set when given a trace. If conditions are added this can be reduced to zero.

A normative model $\mathcal{F}$ corresponds to an AnsProlog program $P_{\mathcal{F}}$ as described in Section 4. All the normal clauses contained in $P_{\mathcal{F}}$ are part of $B$; the only differences involve time points, that are handled in $B$ by means of a finite domain constraint solver. $B$ also contains all the facts in $C$ and $T$ (negated facts are encoded by adding exceptions to the definitions of holdsat and occurred). The set of examples $E$ contains the literals in $O$. Each $H \in ILP(E, B, S)$ represents a possible revision for $P$ and thus for the original normative model.

6 Example

To illustrate the capabilities of the norm learning mechanism, we have developed a relatively simple scenario that, at the same time, is complicated enough to demonstrate the key properties with little extraneous detail.

The active parties—agents—of the scenario each find themselves initially in the situation of having ownership of several (digital) objects—the blocks—that form part of some larger composite (digital) entity—a file. An agent may give a copy of one its blocks in exchange for a copy of another block with the aim of acquiring a complete set of all the blocks. For simplicity, in the situation we analyse here, we assume that initially each agent holds the only copy of a given block, and that is there is only one copy of each block in the agent population. Furthermore, we do not take into account the possibility of exchanging a block for one that the agent already has. We believe that neither of these issues does more than complicate the situation by adding more states that would obscure the essential properties that we seek to demonstrate. Thus, we arrive at a statement of the example: two agents, Alice and Bob, each holding two blocks from a set of four and each having the goal of owning all four by downloading the blocks they miss from the other while sharing, with another agent, the ones it does.
We model this as a simple normative framework, where the brute event [20] of downloading a block initiates several normative events, but the act of downloading revokes the permission of that agent to download another block until it has shared (this the complementary action to download) a block with another agent. Violation of this norm results in the download power being revoked permanently. In this way reciprocity is assured by the normative framework. Initially, each agent is empowered and permitted to share and to download, so that either agent may initiate a download operation.

Fig. 3 shows the AnsProlog representation of the complete normative framework representing this scenario. In the following examples a variety of normative rules will be deliberately removed and re-learned.

\[ \text{Normative and Domain Rules} \]
\[ \text{initiated(filesharing, Agent, Block), I} \leftarrow \text{occurred(filesharing, Agent, Block), I, } \text{holdsat(filesharing, I).} \]
\[ \text{initiated(perm(myDownload, Agent, Block)), I} \leftarrow \text{occurred(myShare(Agent, Block), I, holdsat(filesharing, I).} \]
\[ \text{terminated(pow(fileshearing, myDownload(Agent, Block)), I} \leftarrow \text{occurred(myDownload(Agent, Block), I, holdsat(filesharing, I).} \]
\[ \text{terminated(needsBlock(Agent, Block), I) } \leftarrow \text{occurred(myDownload(Agent, Block), I, holdsat(filesharing, I).} \]
\[ \text{terminated(pow(fileshearing, myDownload(Agent, Block)), I) } \leftarrow \text{occurred(misuse(Agent), I, holdsat(filesharing, I).} \]
\[ \text{terminated(perm(myDownload(Agent, Block)), I) } \leftarrow \text{occurred(myDownload(Agent, Block), I, holdsat(filesharing, I).} \]
\[ \text{occurred(myDownload(AgentA, Block), I) } \leftarrow \text{occurred(download(AgentA, AgentB, Block), I, holdsat(hasBlock(AgentB, Block), I), AgentA = AgentB.} \]
\[ \text{occurred(myShare(AgentB), I) } \leftarrow \text{occurred(download(AgentA, AgentB, Block), I, holdsat(hasBlock(AgentB, Block), I), AgentA = AgentB.} \]
\[ \text{holdsat(pow(filesharing, myDownload(Agent, Block)), i0).} \]
\[ \text{holdsat(pow(filesharing, myShare(Agent)), i0).} \]
\[ \text{holdsat(perm(download(AgentA, AgentB, Block)), i0)).} \]
\[ \text{holdsat(perm(myDownload(Agent, Block)), i0).} \]
\[ \text{holdsat(hasBlock(alice, x1), i0).} \]
\[ \text{holdsat(hasBlock(bob, x3), i0).} \]
\[ \text{holdsat(perm(myShare(Agent)), i0).} \]
\[ \text{holdsat(hasBlock(alice, x2), i0).} \]
\[ \text{holdsat(hasBlock(bob, x4), i0).} \]
\[ \text{holdsat(needsBlock(alice, x3), i0).} \]
\[ \text{holdsat(needsBlock(alice, x4), i0).} \]
\[ \text{holdsat(needsBlock(bob, x1), i0).} \]
\[ \text{holdsat(needsBlock(bob, x2), i0).} \]
\[ \text{holdsat(filesharing), i0).} \]
\[ \text{% fluent rules} \]
\[ \text{holdsat(P, I) } \leftarrow \text{holdsat(P, I), not terminated(P, I), next(I, J).} \]
\[ \text{holdsat(P, I) } \leftarrow \text{initiated(P, I), next(I, J).} \]
\[ \text{occurred(E, X) } \leftarrow \text{evtype(E, ex), observed(X, E).} \]
\[ \text{occurred(viol(E), I) } \leftarrow \text{observed(E, I), not holdsat(perm(E, I), holdsat(filesharing, I), evinst(E, X).} \]
\[ \text{occurred(viol(E), I) } \leftarrow \text{occurred(E, I), evtype(E, inst), not holdsat(perm(E, I), event(viol(E)).} \]

We model this as a simple normative framework, where the brute event [20] of downloading a block initiates several normative events, but the act of downloading revokes the permission of that agent to download another block until it has shared (this the complementary action to download) a block with another agent. Violation of this norm results in the download power being revoked permanently. In this way reciprocity is assured by the normative framework. Initially, each agent is empowered and permitted to share and to download, so that either agent may initiate a download operation.

6.1 Learning Setting
To show how different parts of the formal model can be learned we start from a correct specification and, after deleting some of the rules, we use TAL to reconstruct the missing parts based on a single use case. In our example TAL is set to learn hypotheses of at most three rules with at most three conditions. The choice of an upper bound on
the complexity (number of literals) of the rule ultimately rests on the final user. Alternatively, TAL can iterate on the complexity or perform a best first search that returns increasingly more complex solutions. We use the following mode declarations, $M$:

\[
\begin{align*}
  m_1 : & \text{modeh(terminated}(\text{perm}(\text{myDownload}(+agent, +block)), +\text{instant})). \\
  m_2 : & \text{modeh(terminated}(\text{initiated}(\text{perm}(\text{myDownload}(+agent, +block)), +\text{instant})). \\
  m_3 : & \text{modeb(occurred}(\text{myDownload}(+agent, +block)), +\text{instant})). \\
  m_4 : & \text{modeb(occurred}(\text{myShare}(+agent), +\text{instant})). \\
  m_5 : & \text{modeb(occurred}(\text{myShare}(−agent), +\text{instant})). \\
  m_6 : & \text{modeb}(+(agent) != +(agent)). \\
  m_7 : & \text{modeb(holdsat}(\text{hashblock}(+agent, +block)), +\text{instant})). \\
  m_8 : & \text{modeb(holdsat}(\text{powfilesharing}(\text{myDownload}(+agent, +block)), +\text{instant})).
\end{align*}
\]

The first two mode declarations state that terminate and initiate permission rules for the normative fluent $\text{myDownload}$ can be learned. The other declarations constrain the structure of the body. The difference between $m_3$ and $m_4$ is that the former must refer to the same block as the one in the head of the rule while the latter introduces a possibly different block. $m_8$ is an inequality constraint between agents. In general more mode declarations should be considered (e.g. initiation and termination of all types of fluents should be included) but the revision can be guided by the designer. For example new changes to a stable theory are more likely to contain errors and thus can be isolated in the revision process. The time to compute all the reported hypotheses ranges from 30 to 500 milliseconds on a 2.8 GHz Intel Core 2 Duo iMac with 2 GB of RAM.

The background knowledge $B$ contains the rules in Fig. 3 together with the traces $T$ given in the use cases. $C$ in this example is empty to allow for the demonstration of the most general types of learning.

**Learning a single terminate/initiate rule** We suppose one of the initiate rules is missing from the current specification:

\[
\text{initiated}(\text{perm}(\text{myDownload}(\text{Agent}, \text{Block})), I) \leftarrow \text{occurred}(\text{myShare}(\text{Agent}), I).
\]

The designer inputs the following observed events that show how in a two agent scenario, one of the agents loses permission to download after downloading a block and reacquires it after providing a block for another agent. The trace $T$ looks like:

\[
\begin{align*}
  &\text{observed}(\text{download}(\text{alice}, \text{bob}, x_3), 0), \\
  &\text{observed}(\text{download}(\text{bob}, \text{alice}, x_1), 1).
\end{align*}
\]

The expected output $O$ is:

\[
\begin{align*}
  &\text{not holdsat}(\text{perm}(\text{myDownload}(\text{alice}, x_4)), 1), \\
  &\text{holdsat}(\text{perm}(\text{myDownload}(\text{alice}, x_4)), 2).
\end{align*}
\]

The trace is disfunctional if the expected output is not true in the answer set of $T \cup B$. The ILP task is thus to find a set of rules $H$ within the language bias specified by mode declarations in $M$ such that given the background knowledge $B$ in Fig. 3 and the
given expected output $O$ as conjunction of literals, $O$ is true in the only answer set of $B \cup T \cup H$ (if one exists). $TAL$ produces the following hypotheses:

\[
\begin{align*}
\text{initiated}(\text{perm}(\text{myDownload}(A, \_)), C) & \leftarrow (H_1) \\
\text{occurred}(\text{myShare}(A, C)).
\end{align*}
\]

and

\[
\begin{align*}
\text{terminated}(\text{perm}(\text{myDownload}(\_ \_)), \_ \_). & \quad (H_2) \\
\text{initiated}(\text{perm}(\text{myDownload}(A, \_)), C) & \leftarrow \\
\text{occurred}(\text{myShare}(A, C)).
\end{align*}
\]

The second solution is not the one intended but it still supports the use case. Note that according to current implementation, whenever a fluent $f$ is both initiated and terminated at the same time point, $f$ still holds at the subsequent time point.

**Learning multiple rules** In this scenario two rules are missing from the specification:

\[
\begin{align*}
\text{initiated}(\text{perm}(\text{myDownload}(\text{Agent, Block})), I) & \leftarrow \\
\text{occurred}(\text{myShare}(\text{Agent}, I)). \\
\text{terminated}(\text{perm}(\text{myDownload}(\text{Agent, Block}2)), I) & \leftarrow \\
\text{occurred}(\text{myDownload}(\text{Agent, Block}1), I).
\end{align*}
\]

We use the same $T$ and $O$ as previously. $TAL$ produces the following hypotheses:

\[
\begin{align*}
\text{terminated}(\text{perm}(\text{myDownload}(A, \_)), C) & \leftarrow (H_1) \\
\text{occurred}(\text{myDownload}(A, \_), C). \\
\text{initiated}(\text{perm}(\text{myDownload}(A, \_)), C) & \leftarrow \\
\text{occurred}(\text{myShare}(A, C)). \\
\text{terminated}(\text{perm}(\text{myDownload}(\_ \_)), \_ \_). & \quad (H_2) \\
\text{initiated}(\text{perm}(\text{myDownload}(A, \_)), C) & \leftarrow \\
\text{occurred}(\text{myShare}(A, C)).
\end{align*}
\]

The second solution is consistent with the use case, but the designer can easily discard it, since the rule is not syntactically valid with respect to the normative framework: a fluent can only be terminated as a consequence of the occurrence of an event. Using more advanced techniques for the language bias specification it would be possible to rule out such a hypothesis.

**Learning of undesired violation** We assume the following rule is missing:

\[
\begin{align*}
\text{initiated}(\text{perm}(\text{myDownload}(\text{Agent, Block})), I) & \leftarrow \\
\text{occurred}(\text{myShare}(\text{Agent}, I)).
\end{align*}
\]

This time we provide a different trace $T$:

\[
\begin{align*}
\text{observed}(\text{download}(\text{alice, bob, x3}), 0). \\
\text{observed}(\text{download}(\text{bob, alice, x1}), 1). \\
\text{observed}(\text{download}(\text{alice, bob, x4}), 2).
\end{align*}
\]

As a result of the trace, a violation at time point 2 is implied that the designer knows to be undesired. The expected output is:

\[
\begin{align*}
\text{not occurred}(\text{viol}(\text{myDownload}(\text{alice, x4})), 2).
\end{align*}
\]
occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B), A! = C,\)
\hspace{1cm} (H1)
\text{holdsat}(\text{pow(filesharing, myDownload}(A, E)), B).

occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B), A! = C,\)
\hspace{1cm} (H2)
\text{holdsat}(\text{pow(filesharing, myDownload}(A, E)), B),
\text{holdsat}(\text{hashblock}(A, E), B).

occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B), A! = C,\)
\hspace{1cm} (H3)
\text{holdsat}(\text{pow(filesharing, myDownload}(C, E)), B).

occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B), A! = C,\)
\hspace{1cm} (H4)
\text{holdsat}(\text{hashblock}(A, E), B).

occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B), A! = C,\)
\hspace{1cm} (H5)
\text{holdsat}(\text{hasblock}(A, E), B).

occurred\((\text{myShare}(A, B) \leftarrow \text{download}(C, A, E), B),\)
\hspace{1cm} (H6)
\text{holdsat}(\text{pow(filesharing, myDownload}(C, E)), B).

\textbf{Fig. 4.} Proposals to revise the generate rule

The outcome of the learning consists of the following two possible solutions:

\begin{align*}
\text{initiated}(\text{perm}(\text{myDownload}(A, .), C)) \leftarrow (H_1) \\
\text{occurred}(\text{myShare}(A), C). \\
\text{initiated}(\text{perm}(\text{myDownload}(. .), .)). \quad (H_2)
\end{align*}

that show how the missing rule is derived from the undesired violation. As in the previous scenario the designer can easily dismiss the second candidate.

\textbf{Learning a generate rule} To account for the different type of rules that need to be learned, the language bias is extended to consider learning of generate rules. The new mode declarations are:

\begin{align*}
\text{modeh}(\text{occurred}(\text{myShare}(+\text{agent}), +\text{instant})). \\
\text{modeb}(\text{occurred}(\text{download}(\neg\text{agent}, +\text{agent}, \neg\text{block}), +\text{instant})).
\end{align*}

We use the same trace and expected output as in the previous scenario (three observed events). The following rule is eliminated from the specification:

\begin{align*}
\text{occurred}(\text{myShare}(\text{AgentB}, I) \leftarrow \\
\text{AgentA} = \text{AgentB}, \\
\text{occurred}(\text{download}(\text{AgentA}, \text{AgentB}, \text{Block}), I), \\
\text{holdsat}(\text{hasblock}(\text{AgentB}, \text{Block}), I), \\
\text{holdsat}(\text{pow(filesharing, myDownload}(\text{AgentA}, \text{Block})), I).
\end{align*}

This is the most complicated case for the designer as a set of six different hypotheses are returned by TAL (see Figure 4). Knowing the semantics of the function symbol \text{download}(\text{AgentA}, \text{AgentB}, \text{Block}) as \text{AgentA} downloads from \text{AgentB} the designer should be able to select the most appropriate rule.
7 Related Work

The motivation behind this paper is the problem of how to converge upon a complete and correct normative framework \textit{with respect to the intended range of application}, where in practice these properties may be manifested by incorrect or unexpected behaviour in use. Additionally, we would observe, from practical experience with our particular framework, that it is often desirable, as with much software development, to be able to develop and test incrementally—and regressively—rather than attempt verification once the system is (notionally) complete.

The literature seems to fall broadly into three categories: (a) concrete language frameworks (OMASE, Operetta, InstSuite, MOISE, Islander, OCeAN and the constraint approach of Garcia-Camino (full references to these are currently omitted because of page limitations)) for the specification of normative systems, that are typically supported by some form of model-checking, and in some cases allow for change in the normative structure; (b) logical formalisms, such as [16], that capture consistency and completeness via modalities and other formalisms like [5], that capture the concept of norm change, or [?] and [?]; (c) mechanisms that look out for (new) conventions and handle their assimilation into the normative framework over time and subject to the current normative state and the position of other agents [2, 8]. Essentially, the objective of each of the above is to realize a transformation of the normative framework to accommodate some form of shortcoming. These shortcomings can be identified in several ways: (a) by observing that a particular state is rarely achieved, which can indicate there is insufficient normative guidance for participants, or (b) a norm conflict occurs, such that an agent is unable to act consistently under the governing norms [23], or (c) a particular violation occurs frequently, which may indicate that the violation conflicts with an effective course of action that agents prefer to take, the penalty notwithstanding. All of these can be viewed as characterising emergent [28] approaches to the evolution of normative frameworks, where some mechanism, either in the framework, or in the environment, is used to revise the norms. In the approach taken here, the designer presents use cases that effectively capture their behavioural requirements for the system, in order to ‘fix’ bad states. This has an interesting parallel with the scheme put forward by Serrano and Saugar [30], where they propose the specification of incomplete theories and their management through incomplete normative states identified as “pending”. The framework lets designated agents resolve this category through the speech acts \textit{allow} and \textit{forbid} and scheme is formalised using an action language.

A useful categorisation of normative frameworks appears in [6]. Whether the norms here are ‘strong’ or ‘weak’—the first guideline—depends on whether the purpose of the normative model is to develop the system specification or additionally to provide an explicit representation for run-time reference. Likewise, in respect of the remaining guidelines, it all depends on how the framework we have developed is actually used: we have chosen, for the purpose of this presentation, to stage norm refinement so that it is an off-line (in the sense of prior to deployment) process, while much of the discussion in [6] addresses run-time issues. Whether the process we have outlined here could effectively be a means for on-line mechanism design, is something we have yet to explore.
From an ILP perspective, we employ an ILP system that can learn logic programs with negation (stratified or otherwise). Though recently introduced and in its early stages of development TAL is the most appropriate choice to support this work for two main reasons: it is supported by completeness results, unlike other existing non-monotonic ILP systems ([27], [22]), and it can be tailored to particular requirements (e.g. different search strategies can address performance requirements). The approach presented in this paper is related to other recently proposed frameworks for the elaboration of formal specifications via inductive learning. Within the context of software engineering, [?] has shown how examples of desirable and undesirable behaviour of a software system can be used by an ILP system, together with an incomplete background knowledge of the envisioned system and its environment, to compute missing requirements specifications. A more general framework has been proposed [?] where desirable and undesirable behaviours are generated from counterexamples produced by model checking a given (incomplete) requirements specification with respect to given system properties. The learning of missing requirements has in this case the effect of eliminating the counterexamples by elaborating further the specification.

8 Conclusions and Future Work

We have presented an approach for learning norms and behavioural rules, via inductive logic programming, from example traces in order to guide and support the synthesis of a normative framework. This addresses a crucial problem in normative systems as the development of such specifications is in general a manual and error-prone task. The approach deploys an established inductive logic programming system [12] that takes in input an initial (partial) description of a normative system and use cases of expected behaviours provided by the designer and generates hypothesis in the form of missing norms and behavioural rules that together with the given description explain the use cases. Although the approach presented in this paper has been tailored for learning missing information, it can also be applied to computing revisions over the existing description. In principle this can be achieved by transforming the existing normative rules into defeasible rules with exceptions and using the same ILP system to compute exception rules. These exceptions would in essence be prescriptions for changes (i.e. addition and/or deletion of literals in the body of existing rules) in the current specification. An appropriate refactoring of the defeasible rules based on the learned exception rules would give a revised (non-defeasible) specification. In this case, the revision would be in terms of changes over the rules of a normative framework instead of changes over its belief state, as would be the case if a TMS approach were adopted.

There are several criticisms that can be levelled at the approach as it stands. Firstly, the design language is somewhat unfriendly: a proper tool would have a problem-oriented language, like InstAL/QL [10, 19]. A system designer would then start from an initial description of their normative framework with some use cases and receive automated suggestions of additional norms to include in the framework written in the same high-level language. The machinery described here, based on ASP syntax and ILP formulation, would then be used as a sound “back-end” computation to a formalism familiar to the system designer. Secondly, better control is needed over the rules that are
learned and over the filtering of incorrect rules; at present this depends on specialised knowledge of the learning process. This can to some extent be controlled through careful choice of and limits on the size of use cases—probably involving heuristics—to improve the effectiveness of the learning process in the search for relevant hypotheses and pruning of those potential solutions that cannot be translated back into the canonical form of the normative framework. Despite these issues, we believe we have identified an interesting path for automating and development and debugging of practical normative specifications and perhaps, in the long term, a mechanism for on-line norm evolution.

References

5. Guido Boella, Gabriella Pigozzi, and Leendert van der Torre. Normative framework for normative system change. In Sierra et al. [31], pages 169–176.
16. Christophe Garion, St´ephanie Roussel, and Laurence Cholvy. A modal logic for reasoning on consistency and completeness of regulations. In Boella et al. [4].

47
Norm enforceability in Electronic Institutions?

Natalia Criado¹, Estefania Argente¹, Antonio Garrido¹, Juan A. Gimeno¹, Francesc Igual¹, Vicente Botti¹, Pablo Noriega², Adriana Giret¹

¹ DSIC, Department of Information Systems and Computation, Universitat Politècnica de Valencia, {ncriado,eargente,agarridot,jgimeno,figual,vbotti,agiret}@dsic.upv.es
² IIIA, Artificial Intelligence Research Institute, CSIC, Spanish Scientific Research Council, pablo@iiia.csic.es

Abstract. Nowadays Multi-Agent Systems require more and more regulation and normative mechanisms in order to assure the correct and secure execution of the interactions and transactions in the open virtual organization they are implementing. The Electronic Institution approach for developing Multi-Agent Systems implements some enforceability mechanisms in order to control norms execution and observance. In this paper we study a complex situation in a regulated environment in which the enforceability mechanisms provided by the current Electronic Institutions implementation cannot deal appropriately with norm observance. The analyzed situation is exemplified with a specific scenario of the mWater regulated environment, an electronic market for water-rights transfer. After this example is presented, we extrapolate it to a more generic domain while also addressing the main issues for its application in general scenarios.

1 Introduction

In general, norms represent an effective tool for achieving coordination and cooperation among the members of a society. They have been employed in the field of Multi-Agent Systems (MAS) as a formal specification of a deontic statement that aims at regulating the actions of software agents and the interactions among them. Thus, a Normative MAS (NMAS) has been defined in [3] as follows:

"a MAS organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms and mechanisms to deliberate about norms and detect norm violation and fulfilment."

According to this definition, the norm enforcement problem, faced by this paper, is one of the key factors in NMAS. In particular, this paper faces with the enforcement of norms inside Electronic Institutions (EIs) that simulate real scenarios. EIs [19,22,8] represent a way to implement interaction conventions for agents who can establish commitments in open environments.

When real life problems are modelled by means of EI some of the norms are obtained by giving a computational interpretation to real legislation. In this process we have encountered two main problems:
- **Norm Inconsistency.** Usually the set of laws created by human societies in order to regulate a specific situation are contradictory and/or ambiguous. In particular, there are situations in which there is a general law *(regulative norm [4])* which is controlled by a local law *(procedural norm [4])*). The problem arises when this local law does not ensure compliance of the more general law. This may be due to the existence of different levels of institutions which are working in the same system [11]. Thus, an elaborated process is necessary in order to determine which norms are active in a specific moment and how they are applied. Traditional methods for implementing norms in EI, which are based on the unambiguous interpretation of norms, are not suitable to overcome this problem.

- **Norm Controlling.** Even in absence of a conflict among norms, there is still the problem of norm controlling. Norm enforcement methods inside EI are based on the observation of these activities controlled by norms. In particular, there are norms whose violation cannot be observed since they regulate situations that take place out of the institution boundaries. Thus, violations are only detectable in presence of a conflict among agents.

In this paper we focus on the enforcement of these norms, which cannot be controlled by traditional techniques. Thus, we address the question of enforceability of non-observable norms inside EIs. In order to make more clear and understandable the problem addressed by this paper, it has been exemplified in the mWater scenario [5]. In addition, a first solution for overcoming the mWater concrete problem is shown. In particular, we propose the definition of a grievance scene for allowing normative conflicts to be solved within the mWater institution. However, this solution can be also extrapolated to generic domains.

This paper is structured as follows: the next section provides background on norm implementation, EIs and the implementation of norms inside EIs. Then a concrete example of the problem addressed by this paper is described. Finally, discussion and future works are described.

2 Background

This section firstly reviews the main methods for ensuring norm compliance in MAS and the techniques that can be employed for implementing these methods. Then, a brief description of the Electronic Institution framework is given, as well as a discussion on how norms are implemented and enforced in this framework.

2.1 Norm Implementation in Multiagent Systems

Norms allow legal issues to be modelled in electronic institutions and electronic commerce, MAS organizations, etc. Most of the works on norms in MAS have been proposed from a theoretical perspective. However, several works on norms from an operational point of view have recently arisen, which are focused on giving a computational interpretation of norms in order to employ them in the
design and execution of MAS applications. In this sense, norms must be interpreted or translated into mechanisms and procedures which are meaningful for the society [14]. Methods for ensuring norm compliance are classified into two categories: (i) regimentation mechanisms, which consist in making the violation of norms impossible, since these mechanisms prevent agents from performing actions that are forbidden by a norm; and (ii) enforcement mechanisms, which are applied after the detection of the violation of some norm, reacting upon it.

In a recent work [2], a taxonomy of different techniques for implementing effectively norms is proposed. On the one hand, the regimentation of norms can be achieved by two processes: (i) mediation, in which both the resources and communication channels are accessed through a reliable entity which controls agent behaviours and prevents agents from deviating from ideal behaviour; and (ii) hard-wiring, assuming that the agents’ mental states are accessible and can be modified in accordance with norms. On the other hand, norm enforcement techniques are classified according to both the observer and the enforcer entity. Norms are self-enforced when agents observe their own behaviour and sanction themselves. Thus, norm compliance is both observed and enforced without the need of any additional party. In situations in which those agents involved by a transaction are responsible for detecting norm compliance (i.e. second-party observability) norms can be enforced by: (i) the second-party which applies sanctions and rewards; and (ii) a third entity which is an authority and acts as an arbiter or judge in the dispute resolution process. In the case of third-party observability, two different mechanisms for ensuring norm compliance can be defined according to the entity which is in charge of norm enforcing: (i) social norms are defended by the society as a whole; (ii) in infrastructural enforcement there are infrastructural entities which are authorities in charge of monitoring and enforcing norms by applying sanctions and rewards.

2.2 Electronic Institutions

Electronic Institutions (EI) are computational counterparts of conventional institutions [19, 22, 8]. Institutions are, in an abstract way, a set of conventions that articulate agent interactions [20]. In practice they are identified with the group of agents, standard practices, policies and guidelines, language, documents and other resources —the organization— that make those conventions work. Electronic Institutions are implementations of those conventions in such a way that autonomous agents may participate, their interactions are supported by the implementation and the conventions are enforced by the system on all participants. Electronic institutions are engineered as regulated open MAS environments. These MAS are open in the sense that the EI does not control the agents’ decision-making processes and agents may enter and leave the EI at their own will. EIs are regulated in four ways. First, agents are capable of establishing and fulfilling commitments inside the institution, and those correspond to commitments in the real world. Second, only interactions that comply with the conventions have any consequence in the environment. Third, interactions are
organized as repetitive activities regulated by the institution and, last, interac-
tions, in EIs, are always speech acts.

An EI is specified through: (i) a dialogical framework which fixes the context
of interaction by defining roles and their relationships, a domain ontology and a
communication language; (ii) scenes that establish interaction protocols of the
agents playing a given role in that scene, which illocutions are admissible and
under what conditions; (iii) performative structures that, like the script of a
play, express how scenes are interrelated and how agents playing a given role
move from one scene to another, and (iv) rules of behaviour that regulate how
commitments are established and satisfied.

The IIIA model has a platform for implementation of EIs. It has a graphical
specification language, ISLANDER, in which the dialogical framework, perfor-
mative structures and those norms governing commitments and the pre- and
post-conditions of illocutions are specified [9]. It produces an XML file that is
interpreted by AMELI [10], a middleware that handles agent messages to and
from a communication language, like JADE, according to the ISLANDER spec-
ification [10]. In addition, EIDE [1] includes a monitoring and debugging tool,
SIMDEI that keeps track of all interactions and displays them in different modes.
There is also a tool, aBuilder, that, from the XML specification, generates, for
each role, agent shells that comply with the communication conventions (the
decision-making code is left to the agent programmer).

2.3 Norm Implementation in EI

Norm Regimentation. In AMELI, governors filter the actions of agents, let-
ting them only to perform those actions that are permitted by the rules of
society. Therefore, governors apply a regimentation mechanism, preventing the
execution of prohibited actions and, therefore, preventing agents to violate their
commitments.

This regimentation mechanism employed by governors makes use of a formal-
is based on rules for representing constraints on agent behaviours [13]. This
formalism is conceived as a “machine language” for implementing other higher
level normative languages. More specifically, it has been employed to enforce
norms that govern EIs. The main features of the proposed “machine language”
are: (i) it allows for the explicit definition and management of agent norms (i.e.
prohibitions, obligations and permissions); (ii) it is a general purpose language
not aimed at supporting a specific normative language; (iii) it is declarative and
has an execution mechanism. For implementing this rule system, the Jess tool
has been employed as an inference engine. Jess allows the development of Java
applications with “reasoning” capabilities.

In open systems, not only the regimentation of all actions can be difficult,
but also sometimes it is inevitable and even preferable to allow agents to vi-
olate norms [6]. Reasons behind desirability of norm violations are because it
is impossible to take a thorough control of all their actions, or agents could

\[\text{http://herzberg.ca.sandia.gov/jess/}\]
obtain higher personal benefits when violating norms, or norms may be violated by functional or cooperative motivations, since agents intend to improve the organization functionality through violating or ignoring norms. Therefore, all these situations require norms to be controlled by enforcement mechanisms. Next, works on the enforcement of norms inside EI are described.

**Norm Enforcement.** The enforcement of a norm by an institution requires the institution to be capable of recognizing the occurrence of the violation of the norm and respond to it [14]. Hence, checking activities may occur in several ways: directly, at any time, randomly or with periodical checks, or by using monitoring activities; or indirectly, allowing agents to denounce the occurrence of a violation and then checking their grievances.

Regarding direct norm enforcement, the institution itself is in charge of both observing and enforcing norms. Thus, in this approach there are infrastructural entities which act as norm observers and apply sanctions when a violation is detected. In [17, 12], distributed mechanisms for an institutional enforcement of norms are proposed. In particular, these works propose languages for expressing norms and software architectures for the distributed enforcement of these norms. More specifically, the work described in [17] presents an enforcement mechanism, implemented by the Moses toolkit [16], which is as general (i.e. it can implement all norms that are controllable by a centralized enforcement) and more scalable and efficient with respect to centralized approaches. However, one of the main drawbacks of this proposal is the fact that each agent has an interface that sends legal messages. Since norms are controlled by these local interfaces, norms can be only expressed in terms of messages sent or received by an agent; i.e. this framework does not support the definition of norms that affect an agent as a consequence of an action carried out independently by another agent. This problem is faced by Gaertner et al. in [12]. In this approach, Gaertner et al. propose a distributed architecture for enforcing norms in EI. In particular, dialogical actions performed by agents may cause the propagation of normative positions (i.e. obligations, permissions and prohibitions). These normative propositions are taken into account by the normative level; i.e. a higher level in which norm reasoning and management processes are performed in a distributed manner. In a more recent work, Modgil et al. [18] propose an architecture for monitoring norm-governed systems. In particular, this architecture is formed by trusted observers that report to monitors on states of interest relevant to the activation, fulfillment, violation and expiration of norms. This monitoring system is corrective in the sense that it allows norm violations to be detected and reacting to them.

**Mixed Approaches.** Finally, there are works which employ a mixed approach for controlling norms. In this sense, they propose the usage of regimentation mechanisms for ensuring compliance with norms that preserve the integrity of the application. Unlike this, enforcement is proposed to control norms that cannot be regimented due to the fact that they are not verifiable or their violation
may be desirable. In [7] an example on the mixed approach is shown. In particular, this work shows how norms that define the access to the organization infrastructure are controlled, whereas norms controlling other issues such as work domain norms are ignored. In particular, those norms that define permissions and prohibitions related to the access to the organization are regimented through mediation, whereas obligation norms are enforced following the institutional sanction mechanism.

The ORA4MAS [15] is another well known proposal that makes use of a mixed approach for implementing norms. The ORA4MAS proposal defines artifacts as first class entities to instrument the organisation for supporting agents activities within it. Artifacts are resources and tools that agents can create and use to perform their individual and social activities [21]. Regarding the implementation of norms in the ORA4MAS framework, regimentation mechanisms are implemented in artifacts that agents use for accessing the organization according to the mediation mechanism. Enforcement of norms has been implemented using third party observability, since the detection of norm violations is a functionality provided by artifacts. In addition, norms are enforced by third parties, since there are agents in charge of being informed about norm violations and carrying out the evaluation and judgement of these situations.

However, none of the above mentioned proposals allows norms which regulate activities taking place out of the institution scope to be controlled. In this case, norm compliance is non-observable by the institution and can only be detected when a conflict arises. Thus, in this paper we propose that both a second-party and third-party can observe non-compliant behaviour and start a grievance process which takes place inside the EI. Therefore, in this paper we face the problem of institutional enforcement of norms based on second-party and third-party observability. Next section provides a concrete instantiation of this problem inside a more specific case-study.

3 A concrete sample scenario in the mWater regulated environment

In this section we exemplify the problem of non-regimented norm enforcement in EI with mWater, a regulated MAS application for trading water-rights within a virtual market. In order to get a good understanding of the overall mWater functioning, we first describe the motivation of mWater and present a brief overview of its structure. Afterwards, the sample complex situation for norm enforcement in the current mWater EI implementation is analyzed.

3.1 mWater overall description

In countries like Spain, and particularly in its Mediterranean coast, there is a high degree of public awareness of the main consequences of the scarcity of water and the need of fostering efficient use of water resources. Two new mechanisms
for water management already under way are: a heated debate on the need and feasibility of transferring water from one basin to another, and, directly related to this proposal, the regulation of water banks. $m$Water is an agent-based electronic market of water-rights. Our focus is on demand and, in particular, on the type of regulatory and market mechanisms that foster an efficient use of water while preventing conflicts. The framework is a somewhat idealized version of current water-use regulations that articulate the interactions of those individual and collective entities that are involved in the use of water in a closed basin. The main focus of the work presented in this paper is on the regulated environment, which includes the expression and use of regulations of different sorts: from actual laws and regulations issued by governments, to policies and local regulations issued by basin managers, and to social norms that prevail in a given community of users.

For the construction of $m$Water we follow the IIIA Electronic Institution (EI) conceptual model [1]. For the actual specification and implementation of $m$Water we use the EIDE platform.

---

2 The 2001 Water Law of the National Hidrological Plan (NHP) —‘Real Decreto Legislativo 1/2001, BOE 176’ (see www.boe.es/boe/dias/2001/07/24/pdfs/A26791-26817.pdf, in Spanish)— and its amendment in 2005 regulates the power of right-holders to engage in voluntary water transfers, and of basin authorities to setup water markets, banks, and trading centers for the exchange of water-rights in cases of drought or other severe scarcity problems.
Procedural conventions in the \textit{mWater} institution are specified through a nested performative structure (Fig. 1) with multiple processes. The top structure, \textit{mWaterPS}, describes the overall market environment and includes other performative structures; \textit{TradingHall} provides updated information about the market and, at the same time, users and trading staff can initiate most trading and ancillary operations here; finally, \textit{TradingTables} establishes the trading procedures. This performative structure includes a scene schema for each trading mechanism. Once an agreement on transferring a water-right has been reached it is "managed" according to the market conventions captured in \textit{AgreementValidation} and \textit{ContractEnactment} scenes. When an agreement is reached, \textit{mWater} staff check whether the agreement satisfies some formal conditions and if so, a transfer contract is signed. When a contract becomes active, other right-holders and external stakeholders may initiate a \textit{Grievance} procedure that may have an impact on the transfer agreement. This procedure is activated whenever any market participant believes there is an incorrect execution of a given norm and/or policy. \textit{Grievance} performative structure includes different scenes to address such grievances or for the disputes that may arise among co-signers. On the other hand, if things proceed smoothly, the right subsists until maturity.

3.2 Complex scenario: The registration of water-right transfer agreements

In \textit{mWater} we have three different types of regulations: (i) government norms, issued by the Spanish Ministry of Environment (stated in the National Hydrological Plan); (ii) basin or local norms, defined and regimented by the basin authorities; and (iii) social norms, stated by the members of a given user assembly and/or organization. The interplay among different norms from these three groups brings about complex situations in which there are non-regimented norms and, moreover, the non-compliance of the norm is not observable until a conflict appears. A very critical situation for the reliable execution of \textit{mWater} appears when the following norms apply:

\textit{Government norm - (N0)}: A water-user can use a given volume of water from a given extraction point, if and only if he/she owns the specific water-right or has a transfer agreement that endows him/her.

\textit{Government norm - (N1)}: Every water-right transfer agreement must be registered within the fifteen days after its signing and wait for the Basin Authorities’ approval in order to be executed.

\textit{Local norm - (N2)}: The registration process of a water-right transfer agreement is started voluntarily by the agreement signing parties.

\textit{Social norm - (N3)}: Whenever a conflict appears, a water user can start a grievance procedure in order to solve it.

Sample situation:
Let’s suppose there is a water user \textit{A} who has a water-right \textit{w}_1 and wants to sell it. \textit{A} starts a Trading Table inside the \textit{TradingTables} process (see Fig. 1)
in order to sell $w_1$. The water user $B$ enters the Trading Table and, as a result, there is an agreement $Agr_1$ between $A$ and $B$, by which $B$ buys $w_1$ from $A$ for the period $[t_1, t_2]$, and pays the quantity $p_1$ for such a transfer. $A$ and $B$ belong to Basin$_x$, in which norms $N_1$, $N_2$ and $N_3$ apply. $A$ and $B$ do not register $Agr_1$ due to norm $N_2$ (in other words, $A$ and $B$ do not go to the Agreement Validation scene of Fig. 1). Since there is no mechanism in Basin$_x$ by which water-right $w_1$ is blocked from $A$ after its selling (due to $Agr_1$ is not registered and $w_1$ is still owned by $A$ in time periods not overlapped with $[t_1, t_2]$), $A$ continues to operate in the market. Afterwards $A$ starts a new Trading Table to sell $w_1$ for period $[t_3, t_4]$, with $t_1 < t_3 < t_2$ and $t_4 > t_2$ (the new period $[t_3, t_4]$ is overlapped with $[t_1, t_2]$). In this second Trading Table $A$ and $C$ sign $Agr_2$, by which $A$ sells $w_1$ to $C$ for the period $[t_3, t_4]$ and $C$ pays $p_2$ to $A$. $A$ and $C$ belong to Basin$_x$. In this case $C$ registers $Agr_2$ in the Agreement Validation scene, due to $N_1$ and $N_2$, and obtains the basin approval for executing $Agr_2$. At time $t_3$ (the transfer starting time) $C$ attempts to execute $Agr_2$, but there is no water in the water transportation node, since $B$ is also executing $Agr_1$. At this moment $C$ has a conflict with $B$, and in order to solve it he/she has to start a grievance procedure due to $N_3$ (Grievances performative structure of Fig. 1).

This situation$^3$ is an instantiated example of the one described above, in which there are non-regimented norms whose non-compliance is not observable and cannot be asserted until the conflict appears. The critical situation comes out due to the compliance procedure for agreement registration and second selling of the same water-right is not coercive.

The current development environment of EI we are using does not provide build-in support for non-coercive processes that are defined by non-regimented norms. Moreover, those situations in which it is not possible to observe the non-compliance of a norm until the resulting conflict appears are not supported either. Nevertheless, there are sample scenarios, like $mWater$, in which this behaviour is required. In the following section we analyze the EI implementation we have devised for this complex scenario.

### 3.3 Implementation

In this section our approach to solve the previously described complex scenario in $mWater$ is described.

In order to include norm $N_1$ in the current EI implementation of $mWater$ we have designed the Agreement Validation scene (see Fig. 1) as a successor scene for any Trading Table. When any water user enters this scene, the Market Facilitator verifies the constraint of fifteen days from the agreement statement process related to norm $N_1$. If this constraint is satisfied the water-right transfer agreement is forwarded to the Basin Authority who activates a Normative Reasoning

---

$^3$ The scenario presented in this section happens in practice in Spain, due to the impossibility to monitor all the water transfer negotiations that may take place among the different water users. It can be considered as a loophole in the Spanish regulations. Nevertheless we are interested in modeling it due to its complexity and in order to simulate the "real" behaviour of the basin users.
process in order to approve, or not, the agreement based on the basin normative regulation. If the agreement gets approved it is published in the Trading Hall in order for every water user of the basin to be informed of the transfer agreement.

On the other hand, norm $N2$ is automatically included in the $mWater$ institution due to the EIDE implementation feature by which no participating agent in the electronic institution can be forced to go to a given scene. For the particular $mWater$ example, neither the buyer nor the seller can be forced to go through the transition between the Trading Table scene and the Agreement Validation scene (see Fig. 1). This way, whenever the buyer and/or the seller goes to the Agreement Validation scene he/she starts the scene voluntarily, so norm $N2$ is satisfied.

The implementation of norm $N3$ requires a specific performative structure, named Grievances (Fig. 2), in order to deal with conflict resolution processes.

Finally, the observance of norm compliance is delegated to every water user. Hence, the enforceability of norm $N0$ is delegated to every water user.

Fig. 2 shows the different scenes of the complex Grievances performative structure. In this structure any conflict can be solved by means of two alternative processes (these processes are similar to those used in Alternative Dispute Resolutions and Online Dispute Resolutions [23, 24]). On the one hand, conflict resolution can be solved by means of negotiation tables (Conflict Resolution Negotiation Table performative structure). In this mechanism a negotiation table is created on demand whenever any water user wants to solve a conflict with other/s water user/s, negotiating with them with or without mediator. Such a negotiation table can use a different negotiation protocol, such as face to face, standard double auction, etc. On the other hand, arbitration mechanisms for conflict resolution can also be employed (Arbitration performative structure). In this last mechanism, a jury solves the conflict sanctioning the offenses.

There are three steps in the arbitration process (see Fig. 3). In the first one, the grievance is stated by the plaintive water user. In the second step, the different conflicting parties present their allegations to the jury. Finally, in the last step, the jury, after hearing the dispute, passes a sentence on the conflict. The difference among the two mechanisms for conflict resolution is that the arbitration process is binding, meanwhile the negotiation is not. In this way if
any of the conflicting parties is not satisfied with the negotiation results he/she can activate an arbitration process in order to solve the conflict.

In the previously described complex scenario, when C cannot execute Agr2 (because there is no water in the water transportation node), C believes that B is not complying norm N0. C believes there is a conflict because Agr2 endows him/her to use the water, and moreover, there is no transfer agreement published in the Trading Hall that endows B to do the same. In order to enforce norm N0 and to execute Agr2, C starts a grievance procedure. In this procedure, water users C and B are recruited as conflicting parties and A as third party because he/she is the seller of w1 as stated in Agr2 (Recruiting Conflicting Parties scene of Fig. 2). Let’s assume C chooses as conflict resolution mechanism arbitration, because he/she does not want to negotiate with B. After stating the grievance, C and B present their allegations to the jury. In this process B presents Agr1 by which he/she believes there is fulfillment of norm N0. Nevertheless, in the last arbitration step, by means of a Normative Reasoning function, the jury analyzes the presented allegations and the normative regulations of the basin and deduces that there is an offense. Norm N1 was not complied by B and A, and moreover, A has sold the same water-right twice for an overlapped time period. In this last step, the jury imposes the corresponding sanctions to A and B.

Fig. 4 shows a snapshot of the mWater’s complex scenario implementation running on the AMELI execution environment of EIDE. The implementation we have devised for this complex situation in mWater allows us to solve the described scenario. Moreover, when dealing with this scenario it is possible to observe the limitations of the current EIDE platform for supporting non-observability and enforceability of non-regimented norms. The implementation of mWater we are discussing in this paper is developed with EIDE 2.11, and includes all the components described in previous sections. Moreover, the information model that supports the execution of the EI is developed in MySQL and includes the different conceptual data required for the market execution. Fig. 5 shows a fragment of the relational model in which some elements are depicted such as: basin structure, water-right definition, agreement, and conflict resolution table configuration, among others.

mWater is devised as a simulation tool for helping the basin policy makers to evaluate the behaviour of the market when new or modified norms are ap-

\[\text{Available at http://e-institutions.iiia.csic.es/eide/pub/}\]
plied. To this end, we are working on defining evaluation functions to measure the performance of the market. These measures include the amount of water transfer agreements signed in the market, volume of water transferred, number of conflicts generated, etc. Apart from these straightforward functions we are also working on defining "social" functions in order to assess values such as the trust and reputation levels of the market, or degree of water user satisfaction, among others.

Fig. 4. A snapshot of the mWater electronic institution running on AMELI

Fig. 5. A fragment of the information model of mWater
4 Discussion and closing remarks

In real life problems, in many occasions it is difficult or even impossible to check norm compliance, specially when the violation of the norm cannot be directly observable. In other occasions, it is not only difficult to regiment all actions, but it might be preferable to allow agents to violate norms, since they may obtain a higher personal benefit or they may intend to improve the organization functionality, despite violating or ignoring norms. It is clear that from a general thought and design perspective of an Electronic Institution, it is preferable to define a safe and trustful environment where norms cannot be violated (i.e. norms are considered as hard constraints), thus providing a highly regimented scenario that inspires confidence to their users. However, from a more flexible and realistic perspective, it is appealing to have the possibility for agents to violate norms for personal gain. Although this is a very realistic attribute that humans can have, it eventually leads to corruption and, consequently, the designer may think to rule it out. But again, from a norm enforceability standpoint it is always a good idea to allow this: it does not only make the environment more open and dynamic, but it also provides a useful tool for decision support. In such a thread, we are able to range the set of norms, from a very relaxed scenario to a very tight one, simulate the institution and the agents’ behaviour, and finally analyze when the global performance —in terms of number of conflicts that appear, degree of global satisfaction or corruption, etc.— shows better, which makes it very interesting as a testbed itself [5]. In all these cases, norm enforcement methods are needed, such as second-party and third-party enforcements.

This paper has highlighted the necessity for norm enforceability in Electronic Institutions. Clearly, when the agents and their execution occur outside the boundaries of the institution it is inviable to count on a simple and efficient way to guarantee a norm-abiding behaviour, as the full observability of the whole execution and environment is rarely possible. In other words, norm violations are perfectly plausible (and unfortunately common) and are only detectable in presence of a conflict among agents.

In our mWater scenario, we have proposed an open mechanism that comprises two main principles: (i) the generation of a grievance when one agent detects a conflict, i.e. when an agent denounces the occurrence of a violation; and (ii) an authority entity with the role of arbiter/judge to mediate in the dispute resolution process and being able to apply sanctions. The advantage of this mechanism is twofold. First, it allows different types of grievance, either when it corresponds to the execution of a previous signed (or unsigned) agreement or, simply, when it happens as an occasional event during the habitual execution of the water scenario and its infrastructure use. Second, it provides different ways to deal with grievances, as shown in Fig. 2: (i) in a very formal and strict way by means of an arbitration procedure that relies on a traditional jury, thus applying a third-party enforceability mechanism (with an infrastructure enforcement); or (ii) in a more flexible way that relies on the creation of a conflict resolution negotiation table, which ranges from informal protocols (e.g., face to face) to more formal ones that may need one or more mediators. In this last case, a second-party
enforceability mechanism has been adopted. We have shown that this grievance procedure shows to be effective in the mWater scenario. But despite its origin in the water environment, it can be easily extrapolated to any other real problem modelled by using ELs, which represent the main contributions of this paper.

The underlying idea to deal with norm enforcement in generic domains follows a simple flow, but it needs some issues to be clearly defined. First of all, we require a procedure to activate or initiate a new grievance. This can be done from any type of performative structure similar to the TradingHall of Fig. 1. This operation requires the identification of the agents that will be involved in the grievance itself, so it is essential for all agents to be uniquely identified; that is, we cannot deal with anonymous agents, which is an important issue. Once the grievance has been initiated, we also require a mechanism for recruiting the conflicting parties. Again, this is related to the agents’ identification and the necessity of (perhaps formal) communication protocols to summon all the parties. Note that this step is necessary for any type of dispute resolution, both by negotiation tables and arbitration. And, at this point we have a high flexibility for solving the conflicts, as they can be solved in many ways depending on the type of problem we are addressing at each moment. Analogously to the trading tables that we have in the mWater scenario, we can use general or particular tables to reach an agreement and, thus, solving the conflict, no matter the real problem we have. Finally, it is also important to note that reaching an agreement when solving the conflict does not prevent from having new conflicts that appear from such an agreement, being necessary the initiation of a new grievance procedure and repeating all the operations. Although such new grievances are possible from both the negotiation table and arbitration alternatives, it is common to have situations where the decisions/verdict taken by the arbitration judges are unappealable.

Our current work of research is focused on providing a more thorough specification of this mechanism to enforce norms in ELs, how the conflict resolution tables can be defined and to come up with specialized protocols for these tables. Our final goal is to be able to integrate this behaviour in a decision support system to simulate different agents’ behaviour and norm reasoning to be applied to the mWater and other scenarios of execution.

Acknowledgements

This paper was partially funded by the Consolider programme of the Spanish Ministry of Science and Innovation through project AT (CSD2007-0022, INGENIO 2010), MICINN project TIN2008-06701-C03-03 and by the FPU grant AP-2007-01256 awarded to N. Criado. This research has also been partially funded by the Generalitat de Catalunya under the grant 2009-SGR-1434 and Valencian Prometeo project 2008/051.
References


Towards a Normative BDI Architecture for Norm Compliance

N. Criado¹, E. Argente¹, P. Noriega², and V. Botti³

¹ Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n. 46022 Valencia (Spain)
Email: {ncriado,eargente,vbotti}@dsic.upv.es
² Institut d’Investigació en Intel·ligència Artificial
Consejo Superior de Investigaciones Científicas
Campus de la UAB, Bellaterra, Catalonia (Spain)
Email: pablo@iiia.csic.es

Abstract. Multi-Agent Systems require coordination mechanisms in order to assemble the behaviour of autonomous and heterogeneous agents and achieve the desired performance of the whole system. Norms are deontic statements employed by these coordination mechanisms which define constraints to the potential excesses of agents’ autonomous behaviour. However, norms are only effective if agents are capable of understanding and managing them pragmatically. In this paper, we propose an extension of the BDI proposal in order to allow agents to take pragmatic autonomous decisions considering the existence of norms. In particular, coherence and consistency theory will be employed as a criterion for determining norm compliance.

1 Introduction

The development of network technologies and Internet has made it possible to evolve from monolithic and centralized applications, in which problems are solved by a single component, to distributed applications, in which problems are solved by means of the interaction among autonomous agents. In these systems, the autonomy and heterogeneity of agents make mandatory the definition of coordination mechanisms for ensuring the whole performance of the system. With this aim, social notions, such as organizations, institutions and norms, have been introduced in the design and implementation of distributed systems.

Norms have been defined in distributed systems as regulations or patterns of behaviour established in order to constrain the potential excesses of autonomous agents. The definition of norms for controlling distributed systems requires the development of normative agents. Normative agents [⁸] must be endowed with capabilities for considering norms and deciding which norms to comply with and how to comply with them. The multi-context Graded BDI architecture [⁷] allows agents to reason in uncertain and dynamic environments.
The work presented in [9] is a first effort on the extension of the Graded BDI architecture [7] in order to allow agents to accept norms autonomously. This work focuses on the description of the architecture as a whole but it provides few details about how agents acquire new norms and face with the norm compliance dilemma. In addition, it lacks an elaborated definition of norm and norm dynamics. According to these criticisms, in this paper we propose to revise this architecture in order to allow agents to take norms into account in a more sophisticated way. In particular, here we focus on the application of both Cognitive Coherence Theory [26] and Consistency Theory [2] for reasoning about norm compliance. Coherence is a cognitive theory whose main purpose is the study of how pieces of information influence each other by imposing a positive or negative constraint over the rest of information. Consistency is a logic property which analyses the relationship among a formula and its negation. Our proposal consists on applying deliberative coherence theory for determining which norms are more coherent with respect to the agent’s mental state. In addition, consistency criterion is considered when determining how to comply with norms. Therefore, this paper tries to overlap some of the main drawbacks of the original proposal by means of adding coherence and consistency constraints to the architecture.

This paper is structured as follows: next section describes the background of our proposal; Section 3 provides norm definitions; in Section 4 the normative BDI architecture is explained; the two components in charge of norm management are explained in Sections 5 and 6; Section 7 describes the norm internalization process; and, Section 8 remarks the contributions and future work.

2 Background

Along this section all approaches considered for this work are explained. In particular, the normative multi-context Graded BDI architecture (n-BDI for short) refined in this paper is explained first. Next subsections introduce the basis of consistency and coherence theories.

2.1 BDI architectures for normative agents

Usually, proposals on agent architectures which support normative reasoning do not consider norms as dynamic objects which may be acquired and recognised by agents. On the contrary, these proposals consider norms as static constraints that are hard-wired on agent architectures. Regarding recent proposals on individual norm reasoning, the BOID architecture [4] represents obligations as mental attributes and analyses the relationship and influence of such obligations on agent beliefs, desires and intentions. However, this proposal presents some drawbacks: i) it only considers obligation norms; ii) it considers norms as static entities that are off-line programmed in agents. In relation with this last feature, the EMIL proposal [1] has developed a framework for autonomous norm recognition. Thus, agents would be able to acquire new norms by observing the behaviour of other agents which are situated in their environments. The main disadvantage of EMIL
is that agents obey all recognised norms blindly without considering their own motivations. The multi-context graded BDI agent architecture [7] does not provide an explicit representation of norms. However, it is capable of representing and reasoning with graded mental attitudes, which makes it suitable as a basis for a norm aware agent architecture.

In order to overlap these drawbacks, in [9, 10], the multi-context graded BDI agent architecture [7] has been extended with recognition and normative reasoning capabilities. According to the n-BDI proposal, an agent is defined by a set of interconnected contexts, where each context has its own logic (i.e., its own language, axioms and inference rules). In addition, bridge rules are inference rules whose premises and conclusions belong to different contexts. In particular, an n-BDI agent [9, 10] is formed by:

- **Mental contexts** [6] to characterize beliefs (BC), intentions (IC) and desires (DC). These contexts contain logic propositions such as \((\Psi^\gamma, \delta)\); where \(\Psi\) is a modal operator in \(\{B, D^+, D^-, I\}\) which express beliefs, positive and negative desires and intentions, respectively; \(\gamma \in \mathcal{L}_{DL}\) is a dynamic logic [19] proposition; and \(\delta \in [0, 1]\) represents the certainty degree associated to this mental proposition. For example \((B\gamma, \delta)\) represents a belief about proposition \(\gamma\) of an agent and \(\delta\) represents the certainty degree associated to this belief.

- **Functional contexts** [6] for planning (PC) and communication (CC).

- **Normative contexts** [9] for allowing agents to recognise new norms (RC) and to consider norms in their decision making processes (NC).

- **Bridge Rules** for connecting mental, functional and normative contexts. A detailed description of these bridge rules can be found in [7]. For a more detailed description of normative bridge rules see [9].

Regarding the normative extension of the BDI architecture, the norm decision process consists of the following steps:

1. It starts when the RC derives a new norm through analysing its environment.
2. These norms are translated into a set of inference rules which are included into the NC. The NC is responsible for deriving new beliefs and desires according to the current agent mental state and the inference rules which have been obtained from norms.
3. After performing the inference process for creating new beliefs and desires derived from norm application, the normative context must update the mental contexts.

The original proposal [9, 10] is a preliminary work towards the definition of autonomous norm aware agents capable of making a decision about norm compliance. In this sense, this approach presents several problems and deficiencies. Firstly, the notion of norm is vague and imprecise, in this sense there is not a clear definition of what an abstract norm and a norm instance mean. Regarding the norm recognition process, no details about how the set of abstract norms is updated and maintained are provided. Thus, the RC is seen as a black box that
gives no analysis of how it deals with different types of norms (e.g. social norms, explicit norms created by the institution). Finally, it lacks a more concrete description of how a BDI agent may decide about obeying or not a norm. In this sense, the derivation of positive and negative desires from obligations and prohibitions is too simple. In particular, norms that guide agent behaviours might be in conflict, since they are aimed at defining the ideal behaviour of different roles which may be played by one agent. Besides that, norm compliance decisions should be consistent with the mental state of agents. Therefore, how agents make consistent decisions about norm compliance is the main contribution of the current paper with respect to the original proposal [9].

As a solution to this problem we will employ works on formalisms for ensuring consistency [2] and coherence [26]. In particular, this paper describes how these works are applied for reasoning about norm compliance. Next subsections briefly describe both the proposal of Casali et al. [6] on consistency among graded bipolar desires and the work of Joseph et al. [17] on the formalization of deductive coherence for multi-context graded BDI agents.

### 2.2 Consistency for Graded BDI Agents

In [2] Benferhat et al. made a study of consistency among bipolar graded preferences. Taking this definition of consistency, Casali et al. in [6] proposed several schemas for ensuring consistency among mental graded propositions. In particular, the maintenance of consistency among desires is achieved by means of three different schemas (i.e. $DC_1$, $DC_2$ and $DC_3$) which impose some constraints between the positive and negative desires of a formula and its negation. Thus, $DC_2$ schema (which will be employed in this paper) imposes a restriction over positive and negative desires for the same goal ($(D^+ \gamma, \delta^+\gamma)$ and $(D^-\gamma, \delta^-\gamma)$, respectively). In particular, it claims that an agent cannot desire to be in world more than it is tolerated (i.e. not rejected). Therefore, it determines that:

$$\delta^+\gamma + \delta^-\gamma \leq 1$$

where $\delta^+\gamma$ and $\delta^-\gamma$ are the desirability and undesirability degrees (i.e. the certainty of the positive and negative desire) of proposition $\gamma$, respectively.

### 2.3 Coherence for Graded BDI Agents

In [26] Thagard claims that coherence is a cognitive theory whose main purpose is the study of associations; i.e. how pieces of information influence each other by imposing a positive or negative constraint over the rest of information. According to Thagard’s formalization, a coherence problem is modelled by a graph $g = \langle V, E, \zeta \rangle$; where $V$ is a finite set of nodes representing pieces of information, $E$ are the edges representing the positive or negative constraints among information; each constraint has a weight ($\zeta : E \rightarrow [-1, 1] \setminus \{0\}$) expressing the constraint strength. Maximizing the coherence [26] is the problem of partitioning nodes
into two sets (accepted $A$ and rejected $V \setminus A$) which maximizes the strength of the partition, which is the sum of the weights of the satisfied constraints.

Taking a proof-theoretic approach, Joseph et al. [17] provide a formalization of deductive coherence for multi-context graded BDI agents. Thus, this work proposes a formalization together with mechanisms for calculating the coherence of a set of graded mental attitudes. The main idea beyond this formalism is to consider the inference relationships among propositions belonging to the same context for calculating the weight of coherence and incoherence relationships. Similarly, bridge rules are employed for setting the coherence degree among propositions belonging to different contexts. Details concerning building the coherence graph can be found in [17].

Regarding the relation of coherence with normative decision processes, in [18] Joseph et al. employed coherence as a criterion for rejecting or accepting norms. However, this work is based on a very simple notion of norm as an unconditional obligation. Moreover, this proposal only considers coherence as the one rational criterion for norm acceptance. In addition, the problem of norm conflict has not been faced. Finally the process by which agents’ desires are updated according to norms have also been defined in a simple way without considering the effect of these normative desires on the previous existing desires.

## 3 Norm Notion

Norms have been studied from different fields such as philosophy, psychology, law, etc. MAS research has given different meanings to the norm concept, been employed as a synonym of obligation and authorization [14], social law [20], social commitment [24] and other kinds of rules imposed by societies or authorities.

In this work, we take as a basis the formalization of norms made in [21]. In this proposal a distinction among abstract norms and norm instances is made. An abstract norm is a conditional rule that defines under which conditions obligations, permissions and prohibitions should be created. In particular, the activation condition of an abstract norm defines when an obligation, permission or prohibition must be instantiated. The norm instances that are created out of the abstract norms are a set of active unconditional expressions that bind a particular agent to an obligation, permission or prohibition. Moreover, a norm instance is accompanied by an expiration condition which defines the validity period or deadline of the norm instance.

Following this proposal our definition of both abstract norms and norm instances is provided.

**Definition 1 (Abstract Norm).** An abstract norm is defined as a tuple $n_a = \langle D, A, E, C, S, R \rangle$ where:

- $D \in \{F, P, O\}$ is the deontic type of the norm. In this work obligations ($O$) and prohibitions ($F$) impose constraints on agent behaviours; whereas permissions ($P$) are operators that define exceptions to the activation of obligations or prohibitions;
A is the norm activation condition. It defines under which circumstances the abstract norm is active and must be instantiated.

E is the norm expiration condition, which determines when the norm no longer affects agents.

C is a logic formula that represents the state of affairs or actions that must be carried out in case of obligations, or that must be avoided in case of prohibition norms.

S, R are expressions which describe the actions (sanctions S and rewards R) that will be carried out in case of norm violation or fulfilment, respectively.

Since this work is focused on the norm compliance problem, only those norms addressed to the agent will be taken into account.

**Definition 2 (Norm Instance).** Given belief theory $\Gamma_{BC}$ an abstract norm $n_a = \langle D, A, E, C, S, R \rangle$ is instanciated into a norm instance $n_i = \langle D, C' \rangle$ where:

- $\Gamma_{BC} \vdash \sigma(A)$, where $\sigma$ is a substitution of variables in $A$ such that $A' = \sigma(A)$ and $\sigma(S)$, $\sigma(R)$ and $\sigma(E)$ are fully grounded.
- $C' = \sigma(C)$.

Once the activation conditions of an abstract norm hold it becomes active and several norm instances, according to the possible groundings of the activation condition, must be created. For simplicity, we assume that once a norm is being instantiated then it is fully grounded. In our proposal, the instantiation of activation and expiration conditions are considered by the Norm Instantiation bridge rule (which will be explained in Section 6). Similarly, sanctions and rewards are also considered by this bridge rule in order to decide about convenience of norm compliance. Thus, for simplicity we omit the instantiation of the norm expiration and activation conditions ($\sigma(A)$ and $\sigma(E)$) and the sanction and reward ($\sigma(S)$ and $\sigma(R)$) in the representation of a norm instance.

4 Normative BDI Architecture (n-BDI)

As previously mentioned, the main contribution of this paper is to refine the n-BDI architecture, which was originally proposed in [9, 10], with a more elaborated notion of norm and norm reasoning. In order to design this second version of the n-BDI, the work of Sripada et al. [25] has been considered as a reference. It analyses the psychological architecture subserving norms. In particular, this architecture is formed by two closely linked innate mechanisms: one responsible for norm acquisition, which is responsible for identifying norm implicating behaviour and inferring the content of that norm; and the other in charge of norm implementation, which maintains a database of norms, detects norm violations and generates motivations to comply with norms and to punish rule violators.

The evolution of n-BDI is focused on reasoning about norm compliance and acceptance, so issues related to the detection and reaction to norm violation are beyond the scope of this paper. In this sense, norms affect n-BDI agents in
two ways: i) when a norm is recognised and accepted then it is considered to define new plans; and ii) when accepted norms are active then their instances are used for selecting the most suitable plan which complies with norms. This paper tackles with this last effect of norms. In particular, this paper describes how Deductive Coherence (described in Section 2.3) and Consistency Theory (described in Section 2.2) are applied for reasoning about norm compliance.

The n-BDI refines the normative contexts (described in Section 2.1) according to the norm notions introduced in Section 3. Therefore, Figure 1 shows a scheme of the n-BDI proposed in this paper. In particular, the RC has been redefined as the Norm Acquisition Context (NAC), whereas the NC has been redefined into the Norm Compliance Context (NCC).

![Fig. 1. Normative Extension of the Multi-Context BDI Architecture. Grey contexts and dashed lines (bridge rules) correspond to the basic definition of a BDI agent. The normative extensions are the white contexts and bold lines.]

In this new version of the agent architecture not only the normative contexts have been improved by considering more elaborated normative definitions, but also the norm reasoning process has been extended with consistency and coherence notions. Thus, the norm reasoning process can be described as follows:

1. It starts when the NAC receives information cues for inferring new abstract norms through the Norm Acquisition bridge rule. The NAC carries out an inference process for maintaining the set of abstract norms in force in a specific moment.
2. Once the norm activation conditions hold, abstract norms are instantiated and included into the NCC by means of the Norm Instantiation bridge rule. Then, the NCC carries out an internal process for determining compliance with which of the norm instances. In this sense, not all active norms should be considered when updating the mental state. In this sense, our proposal consists in employing coherence theory as a criterion for determining which norms comply with. Therefore, the coherence maximization process is cal-
culated in order to determine which norm instances are consistent and must be taken into account when updating the desire theory.

3. Then, Norm Internalization bridge rules derive new desires according to the current agent mental state and the set of complied norms, also taking into account consistency considerations. These new desires may help the agent to select the most suitable plan to be intended and, as a consequence, normative actions might be carried out by the agent.

Thus, the norm reasoning process is formed by four different phases: acquisition of norms in force, decision about norm compliance and internalization of norms. Next sections describe each one of these phases in detail.

5 Norm Acquisition (NAC)

The NAC context allows agents to maintain a norm base that contains those norms which are in force in a specific moment (i.e. all norms which are currently applicable). Thus it is responsible for acquiring new norms and deleting obsolete norms; and updating the set of in force norms accordingly. This process can be defined as objective since no motivation or goal is considered in the acquisition process. Thus, agents only take into account their knowledge of the world in order to determine the set of norms which is more likely to be in force.

**NAC Language.** The NAC is formed by expressions such as \((n, \rho)\) where \(n\) is an abstract norm according to Definition 1; and \(\rho \in [0,1]\) is a real value which assigns a degree to this abstract norm. This parameter \(\rho\) can have different interpretations. It can be defined as the reputation of the informer agent in case of leadership-based norm spreading. If norms are inferred by imitation, \(\rho\) might represent the acceptance degree of the norm. In case of utility maximizing approaches, as learning algorithms, it can be defined as the expected utility of the norm.

**Abstract Norm Recognition.** Regarding how new and obsolete norms are recognised, the NAC consists of a computational model of autonomous norm recognition which receives the agent perceptions, both observed and communicated facts, and identifies the set of norms which control the agent environment. Perceptions which are relevant to the norm recognition may be classified into:

- *Explicit normative perceptions.* They correspond to those messages exchanged by agents in which norms are explicitly communicated. Following this approach, several works have focused on analysing the role of leaders in the norm spreading. In particular, these leaders provide normative advices to follower agents when deciding about a norm [27, 22].

- *Implicit normative perceptions.* This type of perceptions includes the observation of actions performed by agents as a way of detecting norms. Since norms are usually supported by enforcing mechanisms such as sanctions and
rewards, the detection of them has been considered as an alternative for acquiring new norms [15]. Other works have proposed imitation mechanisms as a criterion for acquiring new norms. These models are characterized by agents mimicking the behaviour of what the majority of the agents do in a given agent society [28, 5]. Moreover, in [23] researchers have experimented with learning algorithms to identify a norm that maximizes an agent’s utility.

- **Mixed normative perceptions.** There are proposals which consider both explicit and implicit normative perceptions as cues for inferring norms [1].

**Abstract Norm Dynamics.** The set of norms which are in force may change both explicitly, by means of the addition, deletion or modification of the existing norms; and implicitly by introducing new norms which are not specifically meant to modify previous norms, but which change in fact the system because they are incompatible with such existing norms and prevail over them [16]. However, this is a complex issue which is out of the scope of this paper. Works presented at the *Formal Models of Norm Change* are good examples of proposals which provide a formal analysis of all kinds of dynamic aspects involved in systems of norms.

This paper does not focus on the norm acquisition problem and the dynamics of abstract norms. In the following, the NAC will be considered as a *black box* that receives cues for detecting norms as input and generates abstract norms as output.

## 6 Norm Compliance (NCC)

The NCC is the component responsible for reasoning about the set of norms which hold in a specific moment. In this sense the NAC recognizes all norms that are in force, whereas the NCC only contains those norms which are active according to the current situation. The NCC should determine which and how norms will be obeyed and support agents when facing with norm violations. In this sense, the NCC detects norm violations and fulfilments and generates punishing and rewarding reactions. This last issue is over the scope of this paper and will be analysed in future works.

The functionalities carried out by the NCC which are covered by this work are related to three main issues: the NCC is in charge of maintaining the set of instantiated norms which are active; then it considers convenience of norm compliance and determines which norms comply with; and, finally, it derives new desires for fulfilling these norms.

**NCC Language.** The NCC is formed by expressions such as: \((n, \rho)\) where \(n\) is a norm instance according to Definition 2. \(\rho \in [0, 1]\) is a real value which assigns a degree to this norm instance. This parameter can be interpreted as the salience of the norm instance. Its value can be determined according to different

[^3]: http://www.cs.uu.nl/events/normchange2/
criteria such as utility of norm compliance, emotional considerations, intrinsic motivations, etc. In this paper, it is defined with regard to the certainty of norm activation as well as the convenience of norm compliance.

Instantiated norms are inferred by applying instantiation bridge rules to norms when their activation is detected. Next, these normative bridge rules are described in detail.

**Norm Instantiation Bridge Rule.**

\[
\begin{align*}
NAC &: (\langle D, A, E, C, S, R \rangle, \rho), \\
BC &: (B A, \beta_A), BC &: (B \neg E, \beta_E) \\
NCC &: (\langle D, C \rangle, f_{\text{instantiation}}(\theta_{\text{activation}}, \theta_{\text{compliance}}))
\end{align*}
\]

(1)

If an agent considers that an abstract norm \(n_a = (D, A, E, C, S, R)\) is currently active \((B A, \beta_A) \wedge (B \neg E, \beta_E)\) then a new norm instance \(n_i = (D, C)\) is generated. The degree assigned to the norm instance is defined by the \(f_{\text{instantiation}}\) function which combines the values obtained by the \(\theta_{\text{activation}}\) and \(\theta_{\text{compliance}}\) functions.

On the one hand, \(\theta_{\text{activation}}\) combines the evidence about norm activation (i.e. the certainty degrees \(\beta_A, \beta_E\) and \(\rho\)). It can be given a sophisticated definition depending on the concrete application. In this work, it has been defined as the weighed average among these three values, as follows:

\[
\theta_{\text{activation}} = \frac{w_A \times \beta_A + w_E \times \beta_E + w_\rho \times \rho}{w_A + w_E + w_\rho}
\]

If all values are equally weighed, then we obtain that \(\theta_{\text{activation}} = \frac{\beta_A + \beta_E + \rho}{3}\)

On the other hand, \(\theta_{\text{compliance}}\) considers both intrinsic and instrumental motivations for norm compliance. In [11] different strategies for norm compliance from an instrumental perspective over this architecture are described. In particular, they consider the influence of norm compliance and violation on agent’s goals for determining whether the agent accepts the norm. For example, an egoist agent will accept only those norms which benefit its goals (i.e. whose condition is positively desired). In this case:

\[
\theta_{\text{compliance}} = \begin{cases} 
1 & \text{if } \delta^+ \in \Gamma_{DC}; \\
0 & \text{otherwise}
\end{cases}
\]

Finally, values obtained by the \(\theta_{\text{activation}}\) and the \(\theta_{\text{compliance}}\) functions are combined by the \(f_{\text{instantiation}}\):  

\[
f_{\text{instantiation}}(\theta_{\text{activation}}, \theta_{\text{compliance}}) = \frac{w_{\text{activation}} \times \theta_{\text{activation}} + w_{\text{compliance}} \times \theta_{\text{compliance}}}{w_{\text{activation}} + w_{\text{compliance}}}
\]

Again, if these two parameters are equally weighed, then we obtain that 

\[
f_{\text{instantiation}}(\theta_{\text{activation}}, \theta_{\text{compliance}}) = \frac{\theta_{\text{activation}} + \theta_{\text{compliance}}}{2}
\]
This approach relies upon various values such as $w_{\text{compliance}}$, $w_A$, and $w_{\text{activation}}$. The definition of these values is beyond the scope of this paper. In previous works [9, 11, 10], it has been considered that they are defined off-line by the agent designer. However, this solution is static and it does not allow agents to adapt these values according to a changing environment. Thus, this issue will be considered in future works.

6.1 Coherence For Norm Instances

Once Norm Instantiation bridge rule has been applied, it is possible that there is an incoherence between mental propositions. Because of this, a maximizing coherency process is needed in order to determine which propositions are consistent and must be taken into account; and which propositions belonging to the rejection set will be ignored when deriving normative desires.

Since our proposal of agent architecture employs graded logics for representing mental propositions, this work takes as a basis the work described in Section 2.3. As argued before, this work proposes a formalization together with mechanisms for calculating the deductive coherence of a set of graded mental attitudes. Our proposal adapts this work by applying the coherence maximization algorithm to the norm compliance problem. Figure 2 illustrates an overview of the employment of coherence as a criterion for resolving the norm compliance dilemma. As shown by this figure, the normative coherence process considers propositions belonging to the BC, the NCC and NAC. Basically this process takes into account: i) the beliefs that sustain the activation of norms and their relationships among them and other beliefs of the BC; ii) the norm instances and conflict relationships among them; and iii) the abstract norms which have triggered the norm activation. Relationships among propositions belonging to each context are defined by means of inference rules and axioms, whereas coherence connections among propositions of different contexts are defined by means of norm instantiation bridge rules.

By considering coherence we will address three different problems: i) determining norm deactivation; ii) determining active norms in incoherent states and iii) normative conflict resolution. In order to formalize normative incoherence the original proposal of [17] must be extended with extra constraints. Moreover, since we apply the coherence calculation algorithms for improving the normative reasoning, then only those propositions which are relevant to the norm compliance process are taken into account. Next, both the definition of normative coherence constraints for facing with each one of these three questions as well as the determination of relevant propositions is detailed.

Detecting Norm Activation in Incomplete and Inconsistent States. As illustrated in Figure 2, norm instantiation bridge rule (see equation 6) allows norm instances (from NCC) to be connected to beliefs from BC related to their activation and expiration conditions. Norm instantiation bridge rule has as preconditions the belief about the occurrence of the activation condition $A$ and the
negation of the expiration condition $E$. Usually agents do not have an explicit knowledge about the negation of $E$. However, it is possible to infer a certainty degree about $\neg E$ from the certainty degree of $E$. Following this idea, the first step for computing coherence is to calculate the closure under negation of beliefs as follows:

**Definition 3 (Closure of Beliefs under Negation).** Let $\Gamma$ be a finite belief theory presentation using graded formulas. We define the closure of $\Gamma$ under negation as:

$$\Gamma^\sim = \Gamma \cup \{(B\neg \varphi, (1 - \delta)) : (B\neg \varphi, \beta) \notin \Gamma \text{ and } (B\varphi, \delta) \in \Gamma \}$$

Therefore, the closure of a set of beliefs under negation consists on extending this theory by inferring new information from what is actually believed. In particular, if an agent believes that proposition $E$ is true with a certainty degree $\delta$ but it does not have any belief concerning its negation, it is logic to assume that the certainty degree assigned to $\neg E$ should be lower than $(1 - \delta)$. We need to calculate the closure of beliefs under negation in order to detect norm deactivation. In this sense, when the certainty about the expiration condition $E$ increases it can be inferred that the certainty of $\neg E$ decreases even if the agent does not have explicit evidence of it.

In addition we want to define an incoherence relationship among a belief related to a general proposition and its negation. This relationship is defined by means of the addition of an inference rule in the belief context:

$$(BC_1) \ (B\gamma, \beta_{\gamma}), (B\neg \gamma, \beta_{\neg \gamma}) \vdash \ (0, 1 - (\beta_{\gamma} + \beta_{\neg \gamma}))$$

Basically this scheme means that to belief $\gamma$ and $\neg \gamma$ simultaneously is a contradiction $(0)$. The certainty degree of this contradiction is defined in [18] as $1 - (\beta_{\gamma} + \beta_{\neg \gamma})$. 

---

Fig. 2. Usage of coherence as a criterion for resolving the norm compliance dilemma.
One of the main problems of the multi-context BDI architecture is the fact that it does not allow the definition of bridge rules for deleting propositions. In this sense, there is a bridge rule for inferring a new instance of a norm when its activation condition holds. However, it is not possible to create a bridge rule which deletes this instance when the expiration condition holds. In response to this problem, coherence will be used as a criterion for detecting norm deactivation. Moreover, an agent may have beliefs related to the occurrence of both the norm activation and expiration conditions. Thus it should consider all those evidences that sustain the occurrence of the expiration and activation conditions in order to determine the set of norms which are active. In particular, coherence will be used as a criterion for detecting norm activation/deactivation according to the certainty of both the expiration and the activation conditions.

Resolving Normative Conflicts. As previously argued, the above process of normative coherence is useful not only to determine which norms are active but even to resolve a norm conflict. Usually, a norm conflict has been defined in other works as a situation in which something is considered as forbidden and obliged or forbidden and permitted. In our proposal, we define permissions as a normative operator which allows defining an exception to the application of a more general obligation or prohibition norm. Thus, we also consider that norms which define something as forbidden and permitted are also in conflict. However, there is no constraint that represents this type of incoherence. In order to represent incoherence inferred from norm conflicts we add the next inference rules to the NCC:

\[(NCC_1) \quad (\langle O, C \rangle, \rho_O), (\langle F, C \rangle, \rho_F) \vdash (0, -\min(\rho_O, \rho_F))\]
\[(NCC_2) \quad (\langle O, C \rangle, \rho_O), (\langle P, \neg C \rangle, \rho_P) \vdash (0, 1 - (\rho_O + \rho_P))\]
\[(NCC_3) \quad (\langle F, C \rangle, \rho_F), (\langle F, C \rangle, \rho_P) \vdash (0, 1 - (\rho_F + \rho_P))\]

In case of a conflict between a permission and an obligation or a prohibition, the degree of the falsity constant \(0\) is assigned value \(1 - (\rho_O + \rho_P)\) or \(1 - (\rho_F + \rho_P)\), respectively, in a similar way as in \(BC_1\). In case of a conflict among a prohibition and an obligation we define a stronger incoherence by defining the degree of the falsity constant as \(-\min(\rho_O, \rho_F)\).

Selecting Relevant Propositions. Once the coherence graph has been defined and a maximising partition \(\langle A, V \setminus A \rangle\) over this graph has been found following [17], the set of propositions belonging to the NCC (i.e. \(\Gamma_{NCC}\)) is revised in order to consider only the accepted norms:

\[\Gamma_{NCC}' = \Gamma_{NCC} \cap A\]

where \(A\) is the accepted set of norm instances according to the maximizing coherence process [17], i.e. “the most coherent norm instances”.

77
7 Norm Internalization

Regarding works on norm internalization in the MAS community, maybe the most relevant proposal are the works of Conte et al. [8]. According to them, a characteristic feature of norm internalization is that norms become part of the agent’s identity. The concept of identity implies that norms become part of the cognitions of the individual agent. In particular, Conte et al. define norm internalization as a multi-step process, leading from externally enforced norms to norm-corresponding goals, intentions and actions with no more external enforcement. Thus they account for different types and levels of internalization.

In this paper a simplistic approximation to the norm internalization process has been considered. However, it will be object of future work extensions. In particular, we have only considered the internalization of norms as goals. In this sense, the process of norm internalization has been described by the self-determination theory [13] as a dynamic relation between norms and desires. This shift would represent the assumption that internalised norms become part of the agent’s sense of identity. Thus, after performing the coherence process for creating new norm instances, the NCC must update the DC (Figure 1 Norm Internalization Bridge Rules) with the new normative desires. The addition of these propositions into this mental context may cause an inconsistency with the current mental state. As explained in Section 2.2, in [6] three schemas for ensuring consistency among mental graded propositions have been proposed. According to schema $DC_2$, which imposes a restriction over positive and negative desires for a same goal, we have implemented the following inference rule:

$$(DC_2) (D^+\gamma, \delta^+_\gamma), (D^-\gamma, \delta^-_\gamma) \vdash (\overline{0}, 1 - (\delta^+_\gamma + \delta^-_\gamma))$$

Our proposal needs bipolar representation of desires since it is useful when selecting plans to be intended for achieving the desires. In this sense, both negative and positive effects of actions will be taken into account when selecting a plan to be intended. For example, the fact that a plan involves a forbidden action may be considered as a negative effect. Therefore, obligation norms are internalized as positive desires whereas prohibition norms are translated into negative ones. Because of this, $DC_2$ has been considered as a basis for the definition of bridge rules responsible for updating the DC in a consistent way. Next these norm internalization bridge rules are described.

**Norm Internalization Bridge Rules.**

- **Obligation Norm.** According to $DC_2$ schema, bridge rule for updating the DC with the positive desires derived from obligation norms is defined as follows:

$$\begin{align*}
\text{NCC} &: (⟨O, C⟩, ρ), DC : (D^- C, \delta^-), DC : (D^+ C, \delta^+) \\
DC &: (D^+ C, \max(\rho, \delta^+) ), DC : (D^- C, \min(\delta^-, 1 - \max(\rho, \delta^+)) )
\end{align*}$$

$$\text{DC} : (D^+ C, max(\rho, \delta^+) ), DC : (D^- C, min(\delta^-, 1 - max(\rho, \delta^+)))$$

(2)
If an agent considers that the obligation is currently active then a new positive desire will be inferred corresponding to the new norm condition. Thus, the desire degree assigned to the new proposition $C$ is defined as the maximum between the new desirability and the previous value ($\max(\rho, \delta^{+})$). Moreover, the undesirability assigned to $C$ is updated as the minimum between the previous value of undesirability assigned to $\gamma$ ($\delta^{-}$) and its maximum coherent undesirability, which is defined as $1 - \max(\rho, \delta^{+})$.

- **Prohibition Norm.** Bridge rule for updating the DC with negative desires is defined as follows:

\[
NCC : (\langle F, C \rangle, \rho), DC : (D^{-} C, \delta^{-}), DC : (D^{+} C, \delta^{+}) \quad \rightarrow \quad DC : (D^{-} C, \max(\rho, \delta^{-})), DC : (D^{+} C, \min(\delta^{+}, 1 - \max(\rho, \delta^{-})))
\]  

(3)

Similarly to obligation norms, a prohibition related to a condition $C$ is transformed into a negative desire related to the norm condition ($D^{-} C, \max(\rho, \delta^{-})$).

- **Permission Norm.** Finally, permission norms do not infer a positive or negative desire about the norm condition. Permission norms define exceptions to the application of a more general obligation or prohibition norm. As a consequence, they only are defined for creating an incoherence with these more general norms.

8 Conclusion

In this work a previous proposal [9, 10] of a normative BDI architecture has been revised. The first contribution of the current paper is the usage of coherence theory in order to determine what means to follow or violate a norm according to the agent’s cognitions and making a decision about norm compliance. The second contribution of this paper is the employment of consistency notions for updating agent cognitions in response to these normative decisions.

The impact of normative decisions on agent cognitions will be object of future work. In this paper, the norm internalization problem has been faced in a simplistic way by considering only the impact of obeyed norms on agent’s desires. Therefore, in future works the role of both deliberative coherence [26] and emotions on the norm compliance will be analysed. In particular, deliberative coherence, which deals with goal adoption in the context of decision making, will be considered when building plans for obeying or violating norms. In addition, we will work on extending our agent architecture with an explicit representation of emotions which will allow agents to consider phenomena such as shame, honour, gratitude, etc. in their decision making processes.

Due to lack of space, no evaluation or case study has been included here that might provide a more understanding perspective of our proposal. However, works describing the original proposal [10, 11] (neither consistency nor coherence are considered here) provide examples belonging to the m-Water case study. The m-Water [3] is a water right market which is implemented as a regulated open space.
multi-agent system. It is a challenging problem, specially in countries like Spain, since scarcity of water is a matter of public interest. The m-Water framework [12] is a somewhat idealized version of current water-use regulations that articulate the interactions of those individual and collective entities that are involved in the use of water in a closed basin. This is a regulated environment which includes the expression and use of regulations of different sorts: from actual laws and regulations issued by governments, to policies and local regulations issued by basin managers, and to social norms that prevail in a given community of users. For these reasons, we consider the m-Water problem as a suitable case study for evaluating performance of the n-BDI agent architecture, since agents’ behaviour is affected by different sorts of norms which are controlled by different mechanisms such as regimentation, enforcement and grievance and arbitration processes.

Finally, we are working on the implementation of a prototype of the n-BDI architecture. Our aim is to evaluate empirically our proposal through the design and implementation of scenarios belonging to the m-Water case study. In future works, we will make some experiments concerning the flexibility and performance of the n-BDI agent model with respect to simple BDI agents, using the m-Water case study. However, preliminary results of the experimental evaluation of the n-BDI original proposal can be found in [10].

9 Acknowledgments

This work was partially supported by the Spanish government under grants CONSOLIDER-INGENIO 2010 CSD2007-00022, TIN2009-13839-C03-01 and TIN2008-04446 and by the FPU grant AP-2007-01256 awarded to N. Criado.

References

Generating Executable Multi-Agent System Prototypes from SONAR Specifications

Michael Köhler-Buße, Matthias Wester-Ebbinghaus, Daniel Moldt

University of Hamburg, Department for Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
(keohler,weste,moldt)@informatik.uni-hamburg.de

Abstract. This contribution presents the Mulan4Sonar middleware and its prototypical implementation for a comprehensive support of organisational teamwork, including aspects like team formation, negotiation, team planning, coordination, and transformation. Organisations are modelled in Sonar, a Petri net-based specification formalism for multi-agent organisations. Sonar models are rich and elaborated enough to automatically generate all necessary configuration information for the Mulan4Sonar middleware.

1 Introduction

Organisation-oriented software engineering is a discipline which incorporates research trends from distributed artificial intelligence, agent-oriented software engineering, and business information systems (cf. [1, 2] for an overview). The basic metaphors are built around the interplay of the macro level (i.e. the organisation or institution) and the micro level (i.e. the agent). Organisation-oriented software models are particularly interesting for self- and re-organising systems since the system’s organizing principles (structural as well as behavioral) are taken into account explicitly by representing (in terms of reifying) them at run-time.

The following work is based on the organisation model Sonar (Self-Organising Net Architecture) which we have presented in [3, 4]. In this paper we turn to a middleware concept and its prototypical implementation for the complete organisational teamwork that is induced by Sonar.

First of all we aim at a rapid development of our middleware prototype. Therefore we need a specification language that inherently supports powerful high-level features like pattern matching and synchronisation patterns. The second requirement is a narrow gap between the specification and implementation of the middleware prototype. Ideally, middleware specifications are directly executable. As a third requirement, we are interested in well established analysis techniques to study the prototype’s behaviour. As a fourth requirement we want the middleware specifications to be as close as possible to the supported Sonar-model of an organization. Related to this, the fifth requirement results as the possibility to be able to directly generate the middleware specifications from the Sonar-model automatically. The sixth requirement is that we want an easy translation of the prototype into an agent programming language.
Since **SONAR**-models are based on Petri nets we have chosen high-level Petri nets [5] as the specification language for our middleware prototype. This choice meets the requirements stated above: We can reuse **SONAR**-models by enriching them with high-level features, like data types, arc inscription functions etc. Petri nets are well known for their precise and intuitive semantics and their well established analysis techniques, including model checking or linear algebraic techniques. We particularly choose the formalism of **reference nets**, a dialect of high-level nets which supports the nets-in-nets concept [6] and thus allows to immediately incorporate ("program") micro-macro dynamics into our middleware. Reference nets receive tool support with respect to editing and simulation by the **RENEW** tool [7]. Additionally, **RENEW** has been extended by the agent-oriented development framework **Mulan** [8,9], which allows to program multi-agent systems in a language that is a hybridisation of reference nets and Java. We make use of **Mulan** and provide a middleware for **SONAR**-models. Consequently, our middleware is called **Mulan4SONAR** and we present a fully-functional prototype in this paper.

The paper is structured as follows: Section 2 briefly sketches our formal specification language for organisational models, called **SONAR**. Section 3 addresses our **Mulan4SONAR** middleware approach on a rather abstract and conceptual level. It illustrates the structure of our target system: **SONAR**-models are compiled into a multi-agent system consisting of so called position agents, i.e. agents that are responsible for the organisational constraints. Section 4 describes our implemented middleware prototype in detail. It is generated from **SONAR**-models. The middleware serves integration and control of all organisational activities, like team formation, negotiation, team planning, coordination, and transformation. We consider related work in Section 5 before we close the paper with a conclusion in Section 6.

### 2 The Underlying Theoretical Model: **SONAR**

In the following we give a short introduction into our modelling formalism, called **SONAR**. A **SONAR**-model encompasses (i) a data ontology, (ii) a set of interaction models (called **distributed workflow nets**, DWFs), (iii) a model, that describes the team-based delegation of tasks (called **role/delegation nets**), (iv) a network of organisational positions, and (v) a set of transformation rules A detailed discussion of the formalism can be found in [3], its theoretical properties are studied in [4].

In **SONAR** a formal organisation is characterised as a delegation network of sub-systems, called **positions**. Each position is responsible for the execution or delegation of several tasks. Figure 1 illustrates the relationship between the **SONAR** interaction model, the delegation model and the position network – i.e. the aspects (ii) to (iv).

1. The left side of the figure describes the relationship between the positions (here: **broker**, **virtual firm**, **requester**, etc.) in terms of their respective roles (here: **Producer**, **Consumer** etc.) and their associated delegation links. In [3] we have omitted all data-related aspects and transformation rules – i.e. the aspects (i) and (v) – in this figure.
this scenario, we have a requester and two suppliers of some product. Coupling between them is provided by a broker. From a more fine-grained perspective, the requester and one of the suppliers consist of delegation networks themselves. For example, in the case of the virtual firm supplier, we can identify a management level and two subcontractors: firm 1 firm 2. The two subcontractors may be legally independent firms that integrate their core competencies in order to form a virtual enterprise (e.g. separating fabrication of product parts from their assembly). The coupling between the firms constituting the virtual enterprise is apt to be tighter and more persistent than between requester and supplier at the next higher system level, which provides more of a market-based and on-the-spot connection.

SONAR relies on the formalism of Petri nets. Each task is modelled by a place $p$ and each task implementation (delegation/execution) is modelled by a transition $t$. Each task place is inscribed by the set of roles which are needed to implement it, e.g. the set \{Prod, Cons\} for the place in the position requester. Each transition $t$ is inscribed by the DWF net $D(t)$ that specifies the interaction between the roles. In the example we have two inscriptions: $PC$ and $PC3$ where the former is show on the right of Figure 1. Positions are the entities which are responsible for the implementation of tasks.\(^3\) Therefore, each node in $P \cup T$ is assigned to one position $O$.\(^4\)

\(^2\) Note that for this simplified model brokerage is an easy job, since there are only two producers and one consumer. In general, we have several instances for both groups with a broad variety of quality parameters, making brokerage a real problem.

\(^3\) The main distinction between roles and position is that positions – unlike roles – are situated in the organisational network, they implement roles and are equipped with resources.

\(^4\) Organization nets can be considered as enriched organisation charts. Organisation nets encode the information about delegation structures – similar to charts – and also about the delegation/execution choices of tasks, which is not present in charts.
So far we have used only the static aspects of Petri nets, i.e., the graph structure. But SONAR also benefits from the dynamic aspects of Petri nets: Team formation can be expressed in a very elegant way. If one marks one initial place of an organisation net Org with a token, each firing process of the Petri net models a possible delegation process. More precisely, the token game is identical to the team formation process (cf. Theorem 4.2 in [4]). It generates a team net (the team's structure) and a team DWF, i.e. the team's behavior specification.

As another aspect, SONAR-models are equipped with transformation rules. Transformation rules describe which modifications of the given model are allowed. They are specified as graph rewrite rules [10]. As a minimal requirement the rules must preserve the correctness of the given organisational model. In SONAR transformations are not performed by the modeller – they are part of the model itself. Therefore we assume that a SONAR model is stratified by models of different levels. The main idea is that the activities of DWF nets that belong to the level \( n \) are allowed to modify those parts that belong to levels \( k < n \) but not to higher ones.

3 Organisational Position Network Activities

We now elaborate on the activities of a multi-agent system behaving according to a SONAR-model.

3.1 Conceptual Overview

The basic idea is quite simple: With each position of a SONAR-model we associate one dedicated agent, called an organisational position agent (OPA). This is illustrated in Figure 2 where the OPAs associated with a SONAR-model together embody a middleware layer.

![Diagram showing Multi-Agent System and Organisation with positions Alice and Bob, formal and sub-organisations, and channels between them.]

**Fig. 2.** An Organisation as an OPA/OMA Network

An OPA network embodies a formal organisation. An OPA represents an organisational artifact and not a member/employee of the organisation. However, if one fuses all nodes of each position into one single node, one obtains a graph which represents the organisation's chart. Obviously, this construction removes all information about the organisational processes.
each OPA represents a conceptual connection point for an organisational member agent (OMA). An organisation is not complete without its OMAs. Each OMA actually interacts with its OPA to carry out organisational tasks, to make decisions where required. OMAs thus implement/occupy the formal positions.\footnote{Note that from a technical point of view, the OPA network is already a complete MAS. This MAS is highly non-deterministic since a SONAR-model specifies what is allowed and what is obligatory, so many choices are left open. Conceptually, the OPA network represents the formal organisation while the OMAs represent its informal part which in combination describe the whole organisation.} Note that an OMA can be an artificial as well as a human agent. An OPA both enables and constrains organisational behaviour of its associated OMA. Only via an OPA an OMA can effect the organisation and only in a way that is in conformance with the OPA’s specification. In addition, the OPA network as a whole relieves its associated OMAs of a considerable amount of organisational overhead by automating coordination and administration. To put it differently, an OPA offers its OMA a “behaviour corridor” for organisational membership. OMAs might of course only be partially involved in an organisation and have relationships to multiple other agents than their OPA (like Alice and Bob in Figure 2) or even to agents completely external to the organisation (like Alice and Dorothy). From the perspective of the organisation, all other ties than the OPA-OMA link are considered as informal connections.

To conclude, an OPA embodies two conceptual interfaces, the first one between micro and macro level (one OPA versus the complete network of OPAs) and the second one between formal and informal aspects of an organisation (OPA versus OMA). We can make additional use of this twofold interface. Whenever we have a system of systems setting with multiple scopes or domains of authority (e.g. virtual organisations, strategic alliances, organisational fields), we can let an OPA of a given (sub-)organisation act as a member towards another OPA of another organisation. This basically combines the middleware perspective with a holonic perspective (cf. [11]).

### 3.2 Organisational Teamwork

SONAR-models of organisations induce teamwork activities. We distinguish between organisational teamwork activities of first- and of second-order. First-order activities target at carrying out “ordinary” business processes to accomplish business tasks.

- **Team Formation:** Teams are formed in the course of an iterated delegation procedure in a top-down manner. Starting with an initial organisational task to be carried out, successive task decompositions are carried out and sub-tasks are delegated further. A team net according to Section 2 consists of the positions that were involved in the delegation procedure.

- **Team Plan Formation/Negotiation:** After a team has been formed, a compromise has to be found concerning how the corresponding team DWF net (cf. Section 2) is to be executed as it typically leaves various alternatives of going
one way or the other. A compromise is found in a bottom-up manner with respect to the team structure. The "leaf" positions of the team net tell their preferences and the intermediary, inner team positions iteratively seek compromises between the preferences/compromise results of subordinates. The final compromise is a particular process of the team DWF net and is called the team plan.

- **Team Plan Execution**: As the team plan is a DWF net process that describes an interaction between team positions, team plan execution follows straightforward.

- **Hierarchic propagation**: If a holonic approach as illustrated in Figure 2 is chosen, team activities that span multiple organisations are propagated accordingly.

Second-order activities reorganisation efforts.

- **Evaluation**: Organisational performance is monitored and evaluated in order to estimate prospects of transformations. To estimate whether an organisational transformation would improve organisational performance, we introduce metrics that assign a multi-dimensional assessment to a formal organisation. In addition to the Petri net-based specifications of the previous section, there may exist additional teamwork constraints and parameters that may be referred to. How to measure the quality of an organisational structure is generally a very difficult topic and highly contingent. We will not pursue it further in this paper.

- **Organisational Transformations**: As described in Section 2, transformations can either be applied to a formal organisation externally or be carried out by the positions themselves as transformation teams (cf. exogenous versus endogenous reorganisation [12]). In the latter case, transformations are typically triggered by the above mentioned evaluations. But it might also be the case that a new constraint or directive has been imposed and the organisation has to comply.

### 3.3 Organisation Agents

As shown in Figure 2 all the OPAs of an organisation are within the context of an organisation agent which represents the OPA network as a whole. The organisation agent is responsible for the management of the organisational domain data (e.g. customer databases etc.) but also for the management of the organisational meta data which includes the data ontology, the interaction protocols (i.e. the process ontology), and also a representation of the SONAR-model itself. This is illustrated in the top half of Figure 3.

Additionally, the organisation agent is responsible for the network wide framing of the organizational teamwork efforts, i.e. team formation, negotiation, and

---

6 For the time being, we do not address the topic that team plan execution might fail and what rescue efforts this might entail.
team plan execution (as illustrated in the bottom half of Figure 3). The organisation agent is responsible for monitoring the abstract aspects on the teamwork (i.e. the OPA network perspective), while the OPAs are responsible for the concrete decisions (i.e. the OPA perspective).\footnote{Note that the existence of a single agent representing the organisation has not to be confused with a monolithic architecture. The main benefit of the existence of an organisation agent is that it allows to provide a network-wide view on the team activities.} For example, the organisation agent abstractly specifies that during the team formation the OPA $O$ may delegate some task to another agent which must belong to a certain set of OPAs,\footnote{The abstract aspects could as well be implemented by the OPAs themselves and thus be totally distributed. In fact the concurrency semantics of Petri nets perfectly reflects this aspect: In the mathematical sense the processes of an organisation agent are in fact distributed, even if generated from one single net.} but the concrete choice for a partner is left to the OPA $O$ which in turn coordinates its decision with its associated OMA.

In our architecture the concrete choices of the OPAs are framed by the so called team cube (cf. Figure 3). The notation cube is due to the fact that we have three dimension of teamwork: team formation, negotiation, and team plan execution. For each dimension we can choose between several mechanisms. For example in the team formation phase the delegation of tasks to subcontractors can either be implemented by a market mechanism (i.e. choosing the cheapest contractor), by a round-robin scheduling (i.e. choosing contractors in cyclic order), or even by some kind of “affection” between OPAs/OMAs. Given a concrete situation that initiates a teamwork activities, the organisation chooses an appropriate mechanism for

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{organisation.png}
\caption{The Organisation Agent}
\end{figure}
each of the three dimensions. During the execution phase of the team plans the
team cube evaluates the process to improve the assignment of mechanisms.

4 The Mulan4Sonar Middleware

Each position of a Sonar-organisation consists of a formal part (the OPA as an
organisational artifact) and an informal part (the OMA as a domain member). An
organisation together with the OPA network relieves its associated OMA's of a
great part of the organisational overhead by automation of administrative and
coordination activities. It is exactly the generic part of the teamwork activities
from Section 3.2 that is automated by the organisation/OPA network: Team
formation, team plan formation, team plan execution always follow the same
mechanics and OMA's only have to enter the equation where domain actions have
to be carried out or domain-dependent decisions have to be made.

4.1 Compilation of Sonar Specifications into Mulan4Sonar

In the following we demonstrate the compilation of an organisational Sonar-
model into the Mulan4Sonar middleware layer for automated teamwork sup-
port. A Sonar-model is semantically rich enough to provide all necessary in-
formation to allow an automated generation/compilation. The aspects of this
compilation and the resulting prototypical middleware are discussed using the
organisation example introduced above in Figure 1. The prototypical middleware
layer generated from this Sonar-model is specified by a high-level Petri net,
namely a reference net. This is beneficial for two reasons: (1) the translation
result is very close to the original specification, since the prototype directly in-
corporates the main Petri net structure of the Sonar-model; (2) the prototype is
immediately functional as reference nets are directly executable using the open-
source Petri net simulator Renew [7] and we can easily integrate the prototype
into Mulan [8, 9], our developing and simulation system for MAS based on Java
and reference nets. Therefore we have chosen to implement the compiler as a
Renew-plugin.

The plugin implements a compiler that is based on graph rewriting. The com-
piler searches for a net fragment in the Sonar-model that matches the pattern on
the left hand side of a rewrite rule and translates it into a reference net fragment
which is obtained as the instantiation of the rule’s right hand side. An example
rule with the parameter n is given in Figure 4: The rule attaches a place
for the OPA a to the transition. In the final model this place contains the OPA
that represents the position "position name". The rule also adds inscriptions that
describe that OPA a is willing to implement the task t (denoted by the inscrip-
tion a.askimpl(\(n^i\))) and a list of inscriptions a.askPartner(\(n^i, O_i\)) (one for each
\(p_i, 1 \leq i \leq n\)) describing that a delegates the subtask \(p_i\) to the OPA \(O_i\). The
variable \(x\) denotes the identifier of the teamwork process.

We consider teamwork in six phases. For each phase, the original Sonar-model
(in our case the one from Figure 1) is taken and transformation rules generate
Fig. 4. A transformation rule for Phase 1

an executable reference net fragment. For example, the transformation rule from Figure 4 is used for the first phase, selection of team members (see below). Finally, the fragments for the phases 1 to 6 are linked sequentially and the resulting overall net represents the main (organisation-specific) middleware component that is used in the (generic) MULAN4SONAR middleware layer to coordinate the organizational teamwork. The six teamwork phases are the following:

1. Selection of team members: By agents receiving tasks, refining them and delegating sub-tasks, the organisation is explored to select the team agents. This way, a team tree is iteratively constructed but the overall tree is not globally known at the end of this phase.
2. Team assembly: The overall team tree is assembled by iteratively putting sub-trees together. At the end of this phase, only the root agent of the team tree knows the overall team.
3. Team announcement: The overall team is announced among all team member agents.
4. Team plan formation: The executing team agents (i.e. the leaves of the team tree) construct partial local plans related to the team DWF net. These partial plans are iteratively processed by the ancestors in the team tree. They seek compromises concerning the (possibly conflicting) partial plans until the root of the team tree has build a global plan with a global compromise.
5. Team plan announcement and plan localisation: The global team plan is announced among all team member agents. The executing team agents have to localise the global plan according to their respective share of the plan.
6. Team plan execution: The team generates an instance of the team DWF net, assigns all the local plans to it, and starts the execution.

Here, we will only discuss first-order organisational teamwork. However, our MULAN4SONAR middleware approach features a recursive system architecture in order to support reorganization, including second-order activities (a presentation of the whole model can be found in the technical report [13]).

Before the six phases are discussed in more detail, we illustrate how a MULAN multi-agent system that incorporates our MULAN4SONAR middleware layer looks like.
4.2 Multi-Agent System with Mulan4Sonar Middleware Layer

In Section 3, we have described our general vision of a multi-agent system that incorporates Sonar organisations: The formal part of each Sonar organization is explicitly represented by a distributed middleware layer consisting of OPAs for each position and one organisation agent as an additional meta-level entity. In our current prototypical implementation of the Mulan4Sonar middleware layer, the organisation agent is actually not yet fully included, at least not as an agent. Instead, the organisation agent of a Sonar organisation manifests itself in terms of the generated six-phase reference net explained in the previous subsection (together with possible DWF nets). This concept is illustrated in Figure 5.

![Fig. 5. Mulan4Sonar middleware layer in the current prototype](image)

It is shown that the formal part of a Sonar organisation is embodied by the generated middleware net and the position agents that are hosted on the agent places of the net. Here we do not elaborate on the internal structure of the agents as we would have to go into the details of multi-agent system programming with Mulan which is out of the scope of the paper. All OPAs share the same generic OPA architecture (GOPA) that we have presented in [14]. Note that in the current prototype, the OPAs are directly embedded on the agent places of the middleware net. This is justified as they are actually reified parts of the formal organisation and we assume that the whole middleware (and thus the formal organization) is executed on the same Mulan platform. The OMAs however are external agents that have chosen to act as members of the organization. Consequently, they can be hosted on remote platforms and communicate with their respective OPAs via message passing.
For future developments of our Mulan4Sonar middleware we plan to have the organisation agent to be actually realized as a Mulan agent (see Subsection 4.4).

4.3 Explanation of the Six Teamwork Phases

As explained in Subsection 4.1, a Sonar-model of an organisation is compiled into executable reference nets for each of the six teamwork phases. Afterwards, the reference nets for the phases 1 to 6 are combined in one reference net and linked sequentially. This linkage is achieved via synchronisation inscriptions. Thus, the end of a phase is synchronised with the start of the succeeding phase.

The reference nets for the six phases share the same net structure but have different inscriptions. This reflects the fact that all teamwork is generated from the same organisational Sonar-specification, but in different phases different information is needed. Figure 6 shows the generated reference net for the first phase, selection of team members.9

Before any teamwork can occur, the system setup has to be carried out. Six position agents (OPAs) – one for each position – are initialized and registered. The position agents are hosted on the agent places of the generated middleware net. After this step the initialisation is finished and teamwork may ensue.

For our given Sonar-model we have only one position that is able to start a team, namely O1 since it is the only position having a place with an empty preset (i.e. the place p0). Whenever the position agent O1 decides to begin teamwork, it starts the first phase, team member selection. The only possibility for task p0 is to delegate it to O1. Here, O1 has only one implementation possibility for this task, namely t1. This entails to generate the two subtasks p1 and p2. O1 selects the agents these subtasks are delegated to. For p2 there is the only possibility O1 but for p1 there is the choice between O2 and O3. Partner choices occur via the synchronisation askPartner(p, O) between the middleware net and the position agents: Agent a provides a binding for the partner O when the task p has to be delegated. Assume that the agent O1 decides in favour of O3, then the control is handed over to O3 which has a choice how to implement the task: either by t2 or by t3. This decision is transferred between position agents and middleware net via the synchronisation askimpl(t) which is activated by the agent a only if t has to be used for delegation/implementation.

After this iterated delegation has come to an end – which is guaranteed for well-formed Sonar-models – all subtasks have been assigned to team agents and the first phase ends. At this point the agents know that they are team members, but they do not know each other yet. To establish such mutual knowledge the second phase starts.

We cannot cover every phase in detail. The general principle has been shown for phase one, namely enriching the original Sonar model of an organisation with (1) connections to position agents and (2) execution inscriptions along the purpose of the respective teamwork phase.

9 Note that the rule from Figure 4 has been applied several times.
1) Selection of team members (explore organisation)

2) Team assembly
The purpose of the remaining five phases has been covered in Subsection 4.1. Here, we want to cover one technical aspect specifically. The description of the first phase has made clear that it is a top-down phase. Following the delegation relationships of the original SONAR-model, a team tree is built from the root down to the leaves. It is also clear that the second phase has to be a bottom-up phase. The overall team is not yet known to any position agent. Thus, beginning with the leaves of the team tree and the corresponding "one-man sub-teams", sub-teams are iteratively assembled until the complete team is finally known at the root node. Consequently, for the second phase, the direction of the arrows has to be reversed compared to the original SONAR model. Analogous observations hold for the remaining four phases. Phases 3 and 5 are top-down phases while phases 4 and 6 are bottom-up phases.

4.4 Strengths, Weaknesses and Future Work

In this subsection, we give a brief qualitative evaluation of the approach taken in this paper. SONAR is a formal model of organisations based on Petri nets. It is often difficult to initially come up with an approach to deploy formal specifications in a software environment. In the case of the Petri net specifications, one can take advantage of the inherent operational semantics. In this sense, Petri nets often allow for a rapid prototyping approach to go from abstract models (requiring only simple Petri net formalisms) to fully functional, executable models (requiring high-level Petri net formalism, in our case reference nets). Consequently, our first approach was to take a SONAR-specification of an organisation and derive an executable prototype by manually attaching inscriptions and add some auxiliary net elements.

While manually crafting an executable reference net for each specific SONAR-model is of course not worthwhile in the long run, it provided us with very early lessons learned and running systems from the beginning on. The work presented in this paper was the next step. Based on our experiences from the handcrafted prototypes we were able to clearly denominate and devise the transformation rules that were needed for automated generation of executable reference net fragments from SONAR-models.

Consequently, we see the conceptual as well as operational closeness between an underlying SONAR-model and its generated middleware net as a crucial advantage for our fast progress in deploying SONAR-organisations. In addition, formal properties like well-formedness (cf. [3, 4]) of a SONAR-model directly carry over to the implementation level.

However, there are also problems associated with our current approach. Firstly, organisational specifications at run-time are only available in terms of the reference net generated from the underlying SONAR-model. This format is not very suitable for being included in an agent's reasoning processes. Secondly, reorganization efforts are only achieved via a workaround. Changing only particular elements of a reference net at runtime is not inherently supported by our environment. Thus, for a reorganization of an organization, the whole middleware net has to be replaced.
Because of the mentioned problems, we are working on further improving the MULAN4SONAR middleware. Current efforts target at keeping the organizational specification as a more accessible and mutable data structure at the level of the middleware layer. Although it is no longer necessarily represented as a reference net itself, the organisational activities and dynamics allowed by the middleware layer are still directly derived from the underlying Petri net semantics of SONAR.

5 Related Work

Our work is closely related to other approaches that propagate middleware layers for organisation support in multi-agent systems like S-MOISE$^+$ [13], AMELI [16] or TEAMCORE/KARMA [17]. The specifics of each middleware layer depends on the specifics of the organizational model that is supported. What all approaches have in common is that domain agents are granted access to the middleware layer via proxies that constrain, guide and support an agent in its function as a member of the organisation, cf. OrgBox in S-MOISE$^+$, Governor in AMELI, Team Wrapper in TEAMCORE/KARMA. Our organisational position agents, the OPAs, serve a similar purpose. The are coupled with organisational member agents, the OMAs, which are responsible for domain-level actions and decisions.

However, in the case of S-MOISE$^+$ and AMELI, management of organisational dynamics is mainly taken care of by middleware manager agents (the OrgManager for S-MOISE$^+$ and the institution, scene and transition managers for AMELI). The proxies mainly route communication between the domain level agents and the middleware managers. Consequently, middleware management is to some degree centralised. In our case, the OPAs are both proxies and middleware managers. They manage all six phases of organisational teamwork in a completely distributed way. This is quite similar to the function of the Team Wrappers in TEAMCORE/KARMA. The KARMA middleware component can be compared to the organisational agent in our approach. It is a meta-level entity that is responsible for setting up the whole system and for monitoring performance.

In [19], we additionally study the conceptual fit between different middleware approaches (in combination with the organisational models they support) and their application on different levels of a large-scale system of systems.

6 Conclusion

In this paper, we have built upon our previous work SONAR on formalising organisational models for MAS by means of Petri nets [4, 3]. In particular, the paper is dedicated to a prototypical MULAN4SONAR middleware layer that supports the deployment of SONAR-models. As SONAR-specifications are formalised with

---

10 However, in the case of S-MOISE$^+$, the new middleware approach OR4MAS [18] (organizational artifacts for MAS) has been devised, resulting in a more decentralised approach.
Petri nets, they inherently have an operational semantics and thus already lend themselves towards immediate implementation. We have taken advantage of this possibility and have chosen the reference net formalism as an implementation means. Reference nets implement the nets-in-nets concept [6] and thus allow us to deploy SONAR-organisations as nested Petri net systems. The reference net tool RENew [7] offers comprehensive support, allowing us to refine/extend the SONAR specifications into fully executable prototypes.

This leaves us with a close link between a SONAR specification of an organisation and its accompanying MULAN4SONAR middleware support. The structure and behaviour of the resulting software system is directly derived and compiled from the underlying formal model. For example, we have explicitly shown how the organisation net of a formal SONAR-specification can be utilised for the middleware support of six different phases of teamwork. In each phase, the original net is used differently (with different inscriptions and arrow directions). This approach of deploying SONAR-models does not only relieve the developer of much otherwise tedious programming. It also allows to preserve desirable properties that can be proven for the formal model and that now carry over to the software technical implementation.

Finally, although we have introduced the idea of SONAR-organizations acting in the context of other SONAR-organizations, we have not addressed the topic in detail here. We study this subject in [20, 21], but on a more abstract/generic level than SONAR offers. Nevertheless, we have already begun to transfer the results to SONAR.

References

Embodied Organizations: a unifying perspective in programming Agents, Organizations and Environments

Michele Piunti\(^1\), Olivier Boissier\(^2\), Jomi F. Hübner\(^3\), and Alessandro Ricci\(^1\)

\(^1\) Università di Bologna, Italy - \{michele.piunti,a.ricci\}@unibo.it
\(^2\) Ecole des Mines St-Etienne, France - boissier@emse.fr
\(^3\) University of Santa Catarina, Florianópolis, Brazil - jomi@inf.furb.br

Abstract. MAS research pushes the notion of openness related to systems combining heterogeneous computational entities. Typically, those entities answer to different purposes and functions and their integration is a crucial issue. Starting from a comprehensive approach in developing agents, organizations and environments, this paper devises an integrated approach and describes a unifying programming model. It introduces the notion of embodied organization, which is described first focusing on the main entities as separate concerns; and, second, establishing different interaction styles aimed to seamlessly integrate the various entities in a coherent system. An integration framework, built on top of Jason, CArtAgO and Moise (as programming platforms for agents, environments and organizations resp.) is described as a suitable technology to build embodied organizations in practice.

1 Introduction

Agent based approaches consider agents as autonomous entities encapsulating their control, characterized (and specified) by epistemic states (beliefs) and motivational states (goals) which result in a goal oriented behavior. Recently, organization oriented computing in Multi Agent Systems (MAS) has been advocated as a suitable computation model coping with the complex requirements of socio-technical applications. As indicated by many authors \[8, 2, 6\], organizations are a powerful tool to build complex systems where computational agents can autonomously pursue their activities exhibiting social attitudes. The organizational dimension is conceived in terms of functionalities to be exploited by agents, while it is assumed to control social activities by monitoring and changing those functionalities at runtime. Being conceived in terms of human organizations, i.e., being structured in terms of norms, roles and global objectives, this perspective assumes an organizational layer aimed at promoting desired coordination, improving control and equilibrium of social dynamics. Besides, the need for openness and interoperability requires to cope with computational environments populated by several entities, not modelable as agents or organizations, which are supposed to be concurrently exploited by providing functionalities.
supporting agents objectives. These aspects are even more recognized in current ICT, characterized by a massive interplay of self-interested entities (humans therein) developed according to different models, technologies and programming styles. Not surprisingly, recent approaches introduced environment as pivotal dimension in MAS development [22, 14]. Such a multifaceted perspective risks to turn systems into a scattered aggregation of heterogenous elements, while their interplay, as well as their interaction, is reduced to a problem of technological interoperability. To prevent this, besides the different mechanisms and abstractions that must be considered, there is a strong need of binding these elements together in a flexible and clear way.

Providing a seamless integration of the above aspects places the challenge to conceive the proper integration pattern between several entities and constructs. A main concern is agent awareness, namely the need for agents to exhibit special abilities and knowledge in order to bring about organizational and environmental notions—which typically are not native constructs of their architectures [21, 15]. Once the environment dimension is introduced as an additional dimension, a second concern is how to connect in a meaningful way the organizational entities and the environmental ones, thereby (i) how the organization can ground normative measures as regimentation and obligations in environments, and (ii) how certain events occurring in environments may affect the global organizational configuration. These aspects enlighten a series of drawbacks on existing approaches, either on the conceptual model and on the programming constructs to be adopted to build systems in practice.

Taking a programming perspective, this work describes an infrastructural support allowing to seamlessly integrate various aspects characterizing an open MAS. In doing so, the notion of Embodied Organization is detailed, aimed at introducing each element in the MAS as an integral part of a structured infrastructure. In order to reconcile organizations, agents and environments, Embodied organization allows developers to focus on the main entities as separate concerns, and then to establish different interaction styles aimed to seamlessly integrate the various entities in a coherent system. In particular, the proposed approach defines a series of basic mechanisms related to the interaction model:

i. How the agents could profitably interact with both organizational and other environmental entities in order to attain their design objectives;
ii. How the organizational entities could control agent activities and regiment environmental resources in order to promote desired equilibrium;
iii. How environmental changes could affect both organizational dynamics and agents activities;

The rest of the paper is organized as follows: Section 2 provides a survey of situated organization as proposed by existing works. Starting from the description of the basic entities characterizing an integrated perspective, Section 3 presents a unified programming model including agents, organizations and environments. The notion of Embodied Organization is detailed in Section 4, while Section 5 discusses a concrete programming model to implement it in practice.
Finally, Section 6 concludes the paper discussing the proposed approach and future directions.

2 Organizations situated in MAS Environments

Although early approaches in organization programming have not been addressed at modeling environments explicitly, recent trends are investigating the challenge to situate organizations in concrete computational environments. In what follows, a survey on related works is discussed, enlightening strengths and drawbacks of existing proposals.

2.1 Current Approaches

Several agent based approaches allow to implement situated organizations instrumenting computational environments where social interactions are of concern. A remarkable example of situated organization is due to Okuyama et al. [12], who proposed the use of “normative objects” as reactive entities inspectable by agents working in “normative places”. Normative objects can be exploited by the organization to make available information about norms that regulate the behavior of agents within the place where such objects can be perceived by agents. Indeed, they are supposed to indicate obligations, prohibitions, rights and are readable pieces of information that agents can get and exploit in computational environments. The approach envisages a distributed normative infrastructure which is assumed to control emergent dynamics and to allow agents to implicitly interact with a normative institution. The mechanism is based on the intuition that the reification of a particular state in a normative place may constitute the realization of a particular institutional fact (e.g., “being on a car driver seat makes an agent to play the role driver”). This basic idea is borrowed from John Searle’s work on speech acts and social reality [16, 17]. Searle envisaged an institutional dimension rising out of collective agreements through special kind of rules, that he refers as constitutive rules. Those rules constitute (and also regulate) an activity the existence of which is logically dependent on the rules themselves, thus forming a kind of tautology for what a constitutive rule also defines the notion that it regulates. In this view, “being on a car driver seat makes an agent to play the role driver” strongly situate the institutional dimension on the environmental one, both regulating the concept of role adoption and, at the same time, defining it.

Constitutive rules in the form \( X \) counts as \( Y \) in \( C \) are also at the basis of the formal work proposed by Dastani et al. [5]. Here a normative infrastructure (which is referred as “normative artifact”) is conceived as a centralized environment that is explicitly conceived as a container of institutional facts, i.e., facts related to the normative/institutional states, and brute facts, i.e. related to the concrete/ “physical” workplace where agents work. To shift facts from the brute dimension to the normative one the system is assumed to handle constitutive rules defined on the basis of “count-as” and “sanctioning” constructs,
which allows the infrastructure to recast brute facts to institutional ones. The mechanism regulating the application of “count-as” and “sanctioning” rules is then based on a monitoring process which is established as an infrastructural functionality embedded inside the normative system. Thanks to this mechanism, agents behavior can be automatically regulated through enforcing mechanisms, i.e. without the intervention of organizational agents.

A similar approach is proposed in the work by Tinnemeier et al. [20], where a normative programming language based on conditional obligations and prohibitions is proposed. Thanks to the inclusion of the environment dimension in the normative system, this work explicitly grounds norms either on institutional states either on specific environmental states. In this case indeed the normative system is also in charge of monitoring the outcomes of agent activities as performed in the work environment, in so doing providing a twofold support to the organizational dimension and to the environmental one.

With the aim to reconcile physical reality with institutional dimensions, an integral approach has been proposed with the MASQ approach, which introduces a meta-model promoting an analysis and design of a global systems along several conceptual dimensions [19]. The MASQ approach relies on the less recent AGR model, extended with an explicit support to environment as envisaged by the AGRE and AGREEN [1]. Four dimensions are introduced, ranging from endogenous aspects (related to agent’s mental attitudes) to exogenous aspects (related to environments, society and cultures where agents are immersed). In this case, the same infrastructure used to deploy organizational entities is also regulated by precise rules for interactions between agents and environment entities. The resulting interaction model relies on the theory of influences and reactions [9], in the context of which several interaction styles can be established among the heterogenous entities dwelling the system.

Besides conceptual and formal integration, few approaches have accounted a programming approach for situated organizations. By relating situated activities in the workplace, the Brahms platform endows human work practices and allows to represent the relations of people, locations, agent systems, communication and information content [18]. Based on existing theories of situated action, activity theory and distributed cognition, the Brahms language promotes the interplay of intelligent software agents with humans their organizations. A similar idea is provided by Situated Electronic Institutions (SEI) [4], recently proposed as an extension of Electronic Institutions (EI) [7]. Besides providing a runtime management of the normative specification of dialogic interactions between agents, the notion of observability of environment states is at the basis of SEI. They are aimed at interceding between real environments and EI. In this case, special governors, namely modelers, allow to bridge environmental structures to the institution by instrumenting environments with “embodied” devices controlled by the institutional apparatus. Participating agents can, in this case, perform individual actions and interactions (either non message based) while operating upon concrete devices inside the environment. Besides, SEI introduces the notion of staff agents, namely organization aware agents which role is to monitor ongoing
activities performed by agents which are not under the direct control of the institution. Staff agents are then assumed to bridge the gap between participating agents and the institutional dimensions: they typically react to norm violations, possibly ascribing sanctioning and enforcements to disobeying agents. Institutional control is also introduced by the mean of feedback mechanisms aimed at comparing observed properties with certain expected values. On the basis of possible not standard properties detected, an autonomic mechanism specifies how reconfigure the institution in order to re-establish equilibrium.

The ORA4MAS approach [11] proposed a programming model for concretely building systems integrating organizational functionalities in instrumented work environment. In ORA4MAS organizational entities are viewed as artifact based infrastructures. Specialized organizational artifacts (OAs) are assumed to encapsulate organizational functions, which can be exploited by agents to fulfill their organizational purposes. Using artifacts as basic building blocks of organizations, allows agents to natively interact with the organizational entity at a proper abstraction level, namely without being constrained to shape external actions as mechanism-level primitives needed to work with middleware objects. The consequence is that the infrastructure does not rely on a sort of hidden components, but the organizational layer is placed beside the agents as a suitable set of services and functionalities to be dynamically exploited (and created) as an integral part of the MAS work environment. On the other side, ORA4MAS does not provide an explicit support to environmental resources which are not included in the organizational specification. Two types of agents are assumed to evolve in ORA4MAS systems: (i) participating agents, assumed to join the organization in order to exploit its functions (i.e., adopting roles, committing missions etc.), while (ii) organization aware agents, assumed to manage the organization by making changes to its functional and structural aspects (i.e., creating and updating functional schemes or groups) or to make decisions about the deontic events (i.e. norm violations).

2.2 Open Issues and Challenges

Despite the richness of the models proposed for organizations of agents situated in computational environments, many aspects are still under discussion and have still to converge in a shared perspective between the different research lines. In the literature, this variety of approaches have been dealt with separately, each forming a different piece of a global view, with few consideration for how they could fit all together. On these basis, we here enlighten a series of current issues and challenges which our approach, described later on, is going to face with.

Agents/Organisations/Environments Interactions Typically interactions are based on a sub-agentive level, and are founded on protocols and mechanisms, instead on being based on the effective capabilities and functionalities exhibited by the entities involved in the whole system. Different approaches are provided for the interaction model between environment, agents and their organizations. Besides, there is not a clear vision on how environment and organizational en-
tities should support agents in their native capabilities, as for instance the ones related to action and perception.

**Grounding Goals** The computational treatments of goals clashes different approaches once they are referred to agents and their subjective goals, and when they are related to organizations and their global goals. For instance, approaches as MASQ, ORA4MAS describe in a rather abstract terms (i) how the subjective and global goals should be fulfilled in practice; (ii) *which* brute state has to be reached in order to consider a goal as achieved. By considering environments explicitly, either agents and organizations should be able to ground goals to actual environment configurations, thus recognizing the fulfillment of their objectives once the pursued goals have been reached in practice (this approach is adopted, for instance, in [5]). Other approaches, as for instance ORA4MAS [11], do not assume organizations able to automatically detect the fulfillment of global goals in terms of environment configurations.

**Grounding Norms** As for goals, a weak support is provided for grounding norms in concrete application domains, thus allowing to establish how and when a norm has been fulfilled or violated. Furthermore few studies have been addressed at managing norm lifecycle with respect to distributed and (highly) dynamic environments. No agreement is then established on which kind of monitoring and sanctioning mechanisms must be adopted. Some approaches envisage the role of organizational/staff agents [4], other approaches propose the sole automatic regulation provided by a programmable infrastructure [5, 20].

**Agent Awareness** It is not clear which kind of capability, and which grade of awareness, is required for agents to exploit the functionalities provided by the (situated) entities embedding organizational and environmental resources. Related to organizations, some approaches propose agents able to automatically internalize organizational specifications (i.e. MASQ, “normative objects”), other approaches, as (ORA4MAS and SEI) assume agents’ awareness to be *encoded* at a programming level.

**Openness** Concerns about interoperability and openness cross each of the above mentioned aspects. Few approaches account technological integration, for instance with respect to varying agent architectures, protocols and data types. Besides, the described proposals typically focus on a restricted set of interaction styles (i.e. dialogical interactions supported by an institutional infrastructure in SEI, environment mediated interactions in normative objects, an hybrid approach in ORA4MAS).

With the aim to respond the above mentioned challenges, the next sections describe an integrated approach aimed at devising a unified programming model seamlessly integrating agents, organizations and environments.

### 3 Unifying Agents, Organizations and Environments Programming

This section figures out the main elements characterizing an Embodied Organization. It envisages an integrated MAS in terms of societies of agents, envi-
Fig. 1. Structural (a) and Normative (b) specifications for the hospital scenario, represented using the Moise graphical notation.

In order to ease the description, the approach will be sketched in the context of an hospital scenario. It summarizes the dynamics of an ambulatory room, and can be seen as an open system, where heterogenous agents can enter and leave in order to fulfill their purposes. In particular, two types of agents are modeled as organization participants. **Staff agents** (namely physicians and medical nurses) are assumed to cooperate with each other in order to provide medical assistance to visitors. Accordingly, **visitor agents** (namely patients and escorts) are assumed to interact themselves in order to book and exploit the medical examinations provided by the staff.

### 3.1 Organizations

The first considered dimension concerns the organization. We do adopt the Moise model, which allows to specify an organization based on three different dimensions referred as (i) structural, (ii) functional, and (iii) normative.

The Structural Specification (SS) provides the organizational structure in terms of groups of agents, roles and functional relations between roles (links). A role defines the behavioral scope of agents actually playing it, thus providing a standardized pattern of behavior for the autonomous part of the system. An inheritance relation can be specified, indicating roles that extend and inherit properties from parent roles. As showed in Fig. 1 (left), visitor agents can adopt two roles, patient and escort, both inheriting from a visitor abstract role. The doctor role

---

4 We here provide a synthesis of the Moise approach showing the specification of the hospital scenario. For a more detailed description, see [10].
is assumed to be played by a physician. It extends the properties of a more
generic staff role, which is assigned in support and administration activities in-
side the group. Relationships can be specified between roles to define authorities,
communication channels and acquaintance links. Groups consist in a set of roles
and related properties and links. In the hospital scenario escorts and patients
form visit groups, while staff and doctor from staff groups. The specification
allows taxonomies of groups (i.e., escorts and patients forming visit group), and
intra-group links, stating that an agent playing the source role is linked to all
agents playing the target role. Notice that the cardinalities for roles inside a
group are specified, indicating the maximum amount of agents allowed to play
that role. The constraints imposed by the SS allow to establish global properties
on groups, e.g. the well-formedness property means to complain role cardinality,
compatibility, and so on.

The Functional Specification (FS) gives a set of functional schemes specifying
how, according with the SS, various groups of agents are expected to achieve
their global (organizational) goals. The related schemes can be seen as goal
decomposition trees, where the root is a goal to be achieved by the overall group
and the leafs are goals that can be achieved by the single agents. A mission
defines all the goals an agent commits to when participating in the execution of
a scheme and, accordingly, groups together coherent goals which are assigned to a
role in a group. The FS for the hospital scenario (Fig. 2) presents three rehearsed
schemes. The visitor scheme (visitorSch) describes the goal tree related to the
visitor group. It specifies three missions, namely mVisit as the mission to which
each agent joining the visit group has to commit, mPatient as the mission to be
committed by the patient who has to undergo the medical visit, and mPay as
the mission to be committed by at least one agent in the visit group. Notice that
the goals “do the visit” (which is related to the mission mPatient) and “pay
visit” (which is related to the mission mPay) can be fulfilled in parallel. The
monitorSch describes the activities performed by a staff agent. These plans are
aimed at verifying if the activities performed by the visitors follow an expected
outcome, namely if the visitors fulfill the payment committing the mPay mission
(which includes the “pay visit” goal). Finally, the docSch specifies the activities
to which a doctor has to commit, namely to perform the visit to every patient.
Notice that each mission has a further property specifying the maximum amount
of time than an agent has to commit to the mission (“time to fulfill”, or ttf value). The FS also defines the expected cardinality for every mission in the
scheme, namely the number of agents inside the group who may commit a given
mission without violating the scheme constraints.

The Normative Specification (NS) relates roles (as they are specified in the
SS) to missions (as they are specified in the FS) by specifying a set of norms.
Moise norms result in terms of permissions or obligations to commit to a mission.
This allows goals to be indirectly related to roles and groups, i.e. through the
policies specified for mission commitment. Fig. 1 (right) shows the declarative
specification of the norms regulating the hospital scenario, and refers to the
missions described in Fig. 2. “Time to fulfill” (ttf) values refer to the maximum
amount of time the organization expects for the agent to fulfill a norm. For instance, norms $n_1$ and $n_2$ define an obligation for agents playing either patient and escort roles to commit to the $m_{Visit}$ mission. A patient is further obliged to commit to $m_{Patient}$ mission ($n_3$). The norm $n_{10}$ is activated only when the norm $n_6$ is not fulfilled: It specifies an obligation for a doctor to commit the $m_{Staff}$ mission, if no other staff agent is committing to it inside the group. Based on the constraints specified within the SS and FS, the NS is assumed to include an additional set of norms which are automatically generated in order to control role cardinality, goal compliance, deadline of commitments, etc.

The concrete computational entities based on the above detailed specification have been developed based on an extended version of ORA4MAS [11]. This programming approach envisages organizational artifacts (OA) are those non-autonomous computational entities adopted to reify organizations at runtime, thereby implementing the institutional dimension within the MAS. In particular, ORA4MAS adopts two types of artifacts, referred as scheme and group artifacts, which manage the organizational aspects as specified in Moise’s functional, structural and normative dimensions. The resulting system has been referred as Organizational Management Infrastructure (OMI), where the term infrastructure can be understood from an agent perspective: it embeds those organizational functionalities exploitable by agents to participate the organizational activities and to access organization resources possibly exploiting, creating and modifying OAs on the need. Of course, in order to suitably exploit the OMI functionalities, agents need to be equipped with special capabilities and knowledge about the organizational structures, that is what in Subsection 2.2 we refer as agent awareness.

3.2 Environments

As said in Subsection 2.1, the ORA4MAS approach does not support environments besides organizational functionalities. To this end, dually to the OMI, an Environment Management Infrastructure (EMI) is introduced to embed the set of environmental entities aimed at supporting pragmatic functionalities. While artifacts are adopted as basic building blocks to implement the EMI, environments also make use of workspaces (e.g., an Hospital workspace is assumed to contain the hospital infrastructures). Artifacts are adopted in this case to provide a concrete (brute) dimension – at the environment level – to the global system. Workspace are adopted in order to model a notion of locality in terms of an application domain.

As Fig. 2 shows, it is quite straightforward to find a basic set of Environment Artifacts (EA) building the EMI. Taking an agent perspective, the developer here simply imagines which kind of service may be required for the fulfillment of the various missions/goals, thus mapping artifact functionalities to the functional specification given by the Moise FS.

Designing an EMI is thus not dissimilar to instrumenting a real workplace in the human case: (i) to model the hospital room it will be used a specialized hospital workspace, (ii) to automate bookings it will be provided a Desk artifact,
Fig. 2. (Above) A Moise Functional Specification (FS) for the hospital scenario. Schemes are used to coordinate the behavior of autonomous agents. (Below) FS is used to find a set of environmental artifacts, and to map their functionalities in the EMI.

(iii) to finalize visits it will be provided a (program running on an) Surgery Tablet artifact, (iv) to automate payments it will be provided a Billing Machine artifact, and (v) to send fees and bills it will be provided a Terminal artifact.

3.3 Agents

Besides the abstract indication of the different artifacts exploitable at the environment level, the Fig. 2 also shows the actions to be performed by agents for achieving their goals. Thanks to the CArtAgO integration technology, several agent platforms are actually enabled to play in environments: seamless interoperability is provided by implementing a basic set of actions, and related perception mechanisms, allowing agents to interact with artifacts and workspaces [14, 15]. Those actions are directly mapped into artifact operations (functions), or addressed to the workspace: in the case of the EMI, a Jason agent has to perform a joinWorkspace("Hospital") action to enter the room (which is related to the mVisit mission); to book the visit (related to the mVisit mission) the action bookvisit() [artifact_name("Desk")]] has to be performed on the desk artifact, and so on (see Fig. 2, below).

The same semantic mapping agents’ actions into artifact operations is adopted to describe interactions between agents and OMI: e.g., commitMission is an operation that can be used by agents upon the scheme artifact to notify mission commitments; adoptRole (or leaveRole) can be used by an agent upon the group artifact in order to adopt (leave) a given role inside the group, etc.

Fig. 3 (left) shows a global picture of the resulting system. As showed, agents fulfill their goals and coordinate themselves by interacting with EMI artifacts,
while staff agents, which we assume as special agents aware of organizational functionalities, can directly interact with the OMI. Both these dimensions are an integral part of the global infrastructure and, most important, can be dynamically exploited by agents to serve their purposes. From an agent perspective, the whole system can be understood as a set of facts and functions, which are exploited, from time to time, to the organizational and environmental dimensions. Through artifacts, the global infrastructure provides observable states, namely information readable by agents for improving their knowledge. Artifacts also provide operations, namely process based functionalities, aimed at being exploited by agents for externalizing activities in terms of external actions. Thus, the epistemic nature of observable properties can be addressed to the informational dimension of the whole infrastructure, while the pragmatic nature of artifact operations is assumed to cover the functional dimension.

4 Embodied Organizations

As far as the global system is conceived, EMI and OMI are situated side by side inside the same work environment, but they are conceived as separated systems. They are assumed to face distinct application domains, the former being related to concrete environment functionalities and the latter dealing specifically with organizational ones. The notion of Embodied Organization provides a more strict integration: it further identifies and implements additional mechanisms and conceives a unified infrastructure enabling functional relationships between EMI and OMI. As some of the approaches discussed in Section 2, we theoretically found this relationship on Searle’s notion of constitutive rules. Differently from other approaches, we ground the notion of Embodied Organization

---

Fig. 3. (Left) Global view of the system presents an open set of agents at work with infrastructures managing Environment and Organization. Functional relationships between EMI and OMI are established by count-as and enact rules. (Right) Meta-model for Organizational Embodied Rules, used to implement count-as and enact rules.
on a concrete programming model, as the one who lead us to the implementation of EMI and OMI. As explained below, Embodied Organizations rely on a revised management of events in CArtAgO, and can be specified by special programming constructs referred as Emb-Org-Rules.

4.1 Events

A crucial element characterizing Embodied Organizations is given by the renewed workspace kernel based on events. Events are records of significant changes in the application domain, handled at a platform level inside CArtAgO. They are referred to both state and processes to represent the transitions of configurations inside workspaces. Each event is represented by a type-value pair \((ev_t, ev_v)\): Event type indicates the type of the event (i.e., join_req indicating agents joining workspace, op_completed indicating the completion of an artifact operation, signal indicating events signalled within artifact operation execution, and so on); Event value gives additional information about the event (i.e., the source of the event, its informational content, and so on). Due to the lack of space, the complete list of events, together with the description of the mechanism underlying event processing, can not be described here. The interested reader can find the complete model, including the formal transition system, in [13]. We here emphasize the relevance of events, which have the twofold role (i) to be perceived or triggered by agents (i.e. focusing/using artifacts) and (ii) to be collected and ranked within the workspace in order to trace the global dynamic of the system.

4.2 Embodied Organization Rules

While the former role played by events refers to the interaction between agents and artifacts, the second role is exploited to identify, and possibly govern, intra-workspace dynamics. On such a basis, the notion of Embodied Organization refers to the particular class of situated organization structured in terms of artifact based infrastructures and governed by constitutive rules based on workspace events. Events are originated within the infrastructure, being produced by environmental and organizational entities. Computing constitutive rules is realized by Emb-Org-Rule, which consist of a programmable constructs “gluing” together organizational and environmental dimensions. An abstract model of this process is shown by the dotted arrows between EMI and OMI in Fig. 3 (right). Structures defining Emb-Org-Rule refer to count-as and enact relations. Count-as rules state which are the consequences, at the organizational level, for an event generated inside the overall infrastructure. They indicate how, since the actions performed by the agents, the system automatically detects relevant events, thus transforming them to the application of a set of operators aimed at changing the configuration of the Embodied Organization. In so doing, either relevant events occurring inside the EMI (possibly triggered by agents actions), either events occurring in the context of the organization itself (OMI) can be vehicled to the institutional dimension: these events can be further translated in
the opportune institutional changes inside the OMI, that is assumed to update accordingly. **Enact rules** state, for each institutional event, which is the control feedback at the environmental level. Hence, **enact** rules express how the organizational entities automatically control the environmental ones. The use of enact rules allows to exploit organizational events (i.e. role adoption, mission commitment) in order to elicit changes in the environment.

5 Programming Embodied Organizations

Embodied Organizations enable a unified perspective on agents, organizations and environments by conceiving an interaction space based on a twofold infrastructure governed by events and constitutive rules (**Emb-Org-Rules**). In this section examples of programming such rules are discussed.

**Programming Count-as Rules** According to the Moise FS previously defined, the organization expects that an agent $va_{id}$ joining the hospital workspace is assumed to play the role visitor, which purpose is to book a medical visit and possibly achieve it. Thus, an event $join\_req$, $\langle va_{id}, t \rangle$, dispatched once an agent $va_{id}$ tries to enter the workspace, from the point of view of the organization “count-as” creating a new position related to the visit group. Making the event $join\_req$ to “count as” $va_{id}$ adopting the role visitor, is specified by the first rule in Table 1 (left); it states that since an event signalling that an agent $Ag$ is joining the workspace, an **Emb-Org-Rule** must be applied to the system. The body of the rule specifies that two new instances of organizational artifacts related to the visit group will be created using the **make** operator. In this case the new
artifacts will be identified by visitorGroupBoard and visitorSchBoard. The following operator constitutes the new role inside the group: apply acts on the visitorGroupBoard artifact just created by automatically making the agent Ag to adopt the role patient. Finally, once the adopt role operator succeeds, the last operator includes the agent Ag in the workspace.

In the above described scenario, the effect of the application of the rule provides an institutional outcome to the joinWorkspace actions. Besides joining the workspace, a sequence of operators is applied establishing what this event means in organizational terms. When the effects of the role-adoptions are committed, as previously described, a new event is generated by the group board: (op_completed, {"visitorGroupBoard", vaid, adoptRole, patient}). For the organization, such an event may “count-as” committing to mission mPat on the visitorSchBoard. This relation is specified by the second rule in Table 1, where a commitMission is applied to the visitorSchBoard for the mission mPat. Similarly, an event (ws_lefted, (vaid,t)), signalling that the visitor agent has left the workspace, from an organizational perspective “count-as” leaving the role patient. This relation is specified by the first rule in Table 1 (right), where a leaveRole is applied to the visitorGroupBoard for the role patient. At the same time, an event like (op_completed, (BillingMachine, vaid, pay, t)) signals that a visitor agent has successfully finalized the pay operation upon the billing machine. Such an event “count-as” having achieved the goal pay visit on the visitorSchBoard (second rule in Table 1, right). Finally, an event (op_completed, (Terminal, said, sendFee, t)), signalling that a staff agent has successfully used the terminal to send the fee to a given patient, “count-as” having achieved the goal send fee (third rule in Table 1, right).

Programming Enact Rules Enact effects are defined to indicate how, from the events occurring at the institutional level, some control feedback can be applied to the environmental infrastructure. As far as the execution of the operations is conceived in CArtAgO, the OMI automatically dispatches events signalling ongoing violations. Violations are thus organizational events which may suddenly elicit the application of some enact rule used to regiment the environment.

In Table 2, a regimentation is installed by the organization thanks to the enact rule stating that an event (signal, (visitorGroupBoard, role_cardinality, ∅, t)) signalled by the visitorGroupBoard indicates the violation for the norm role_cardinality. The related enact rule is given in Table 2 (left), where the reaction to this event is specified in order to disable the book operation on the desk artifact, for all the agents inside the workspace. The absence of any parameter related to agent identifier in the disable("Desk", bookVisit) operator makes the disabling to affect the overall set of agents inside the workspace. Similarly, violating the obligation imposed to the staff agent to fulfill sanctioning and rewarding missions elicits the scheme board assigned to the monitorSch to signal the event (signal, (monitorSchBoard, goal_non_compliance, obligation(Ag, ngoal(monitorSch, mRew, send_bill), achieved(monitorSch, send_bill, Ag), TTF), t)). This event is generated thanks to a special norm (called goal_non_compliance) which is automatically generated since the Moise specification and stored in-
side the OMI. Due to the enact rule specified in Table 2 (right), this causes the exclusion for the $Ag$ agent from the hospital workspace.

6 Conclusion and Perspectives

In this paper Embodied Organizations have been introduced as a unified programming model promoting a seamless integration of environmental and organizational dimensions of a MAS. A series of responses to the challenges envisaged in Subsection 2.2 could be listed: Infrastructures. Either environmental and organizational entities are implemented in concrete infrastructures instrumenting workspaces, decentralized in specialized artifacts which serve informational and operational functions. Interaction. The approach establishes a coherent semantic for agent-infrastructure interactions, Embodied Organizations define functional relationships between the heterogenous entities at the basis of organizations and environments. These are placed in terms of programmable constructs (Emb-Org-Rules), governed by workspace events and inspired by Searle’s notion of constitutive rules. Goals and Norms. Implementing organizations in concrete environments allows to deal explicitly with goals and norms, which fulfillment can be structurally monitored and promoted at the organizational level through the use of artifacts. Awareness. Embodied Organizations are aimed to fit the work of agents and accordingly to allow them to externalize pragmatic and organizational activities. The use of Emb-Org-Rule automates and promotes specific organizational patterns, to which agents may effortlessly participate simply by exploiting environmental resources. Artifacts can be used in goal oriented activities, and, most important, without the need to be aware of organizational notions like roles, norms, etc. Openness. Technological interoperability is ensured at a system level, by providing mechanisms for agent-artifact interactions which are based on a coherent semantic. Besides, several interaction styles can be established at an application level, being agents mediated by infrastructures which can be modified, replaced and created on the need.

Future work will be addressed at covering missing aspects, such as the dialogical dimension of interactions, and the inclusion of real embodied entities in the system (i.e., humans, robots, etc.). An important objective is the definition of a general purpose approach, towards the full adoption of the proposed model in the context of concrete application domains and mainstream agent oriented programming.

References

2. Olivier Boissier, Joni Fred Hübner, and Jaime Simão Sichman. Organization Oriented Programming: From Closed to Open Organizations. In Engineering Societies


Group Intention is Social Choice with Commitment

Guido Boella\textsuperscript{1}, Gabriella Pigozzi\textsuperscript{2}, Marija Slavkovik\textsuperscript{2}, and Leendert van der Torre\textsuperscript{2}

\textsuperscript{1} guido@di.unito.it
\textsuperscript{2} \{gabriella.pigozzi, marija.slavkovik, leon.vandertorre\}@uni.lu

Abstract. A collaborative group is commonly defined as a set of agents, which share information and coordinate activities while working towards a common goal. How do groups decide which are their common goals and what to intend? According to the much cited theory of Cohen and Levesque, an agent intends $g$ if it has chosen to pursue goal $g$ and it has committed itself to making $g$ happen. Following the same line of reasoning, a group intention should be a collectively chosen goal with commitment. The literature often considers a collective goal to be one of those individual goals that are shared by all members. This approach presumes that a group goal is also an individual one and that the agents can act as a group if they share the beliefs relevant to this goal. This is not necessarily the case. We construct an abstract framework for groups in which common goals are determined by social choice. Our framework uses judgment aggregation to choose a group goal and a multi-modal multi-agent logic to define commitment and revision strategies for the group intentions.

1 Introduction

An agent acts according to its beliefs and intentions. According to what does a group act? We would expect that in order for groups to act, jointly or by coordinating their activities, they need to establish what to believe i.e., form epistemic attitudes, and what to aim for, i.e., to form motivational attitudes. In existing frameworks for collaborative activities \cite{9, 12, 16} and group decision-making protocols \cite{10}, the formation of group attitudes is defined only for a specific type of groups. These groups consist of agents that engage in pursuing a group goal only when the members have the same beliefs regarding this goal, or when they are successful in reaching an agreement on a given set of beliefs.

Consider a group of robots in charge of office building maintenance. One candidate group goal for them is to clean the meeting room. The decision rule to adopt the goal is that the room needs to be cleaned when the following conditions (reasons) are met: the floors are dirty or the garbage bin is full, and people do not occupy the room. To decide whether to pursue this goal, the robots need to decide if the reasons to adopt the goal are true. The robots cannot check whether the room is occupied or what state it is in. Hence, to estimate the state of affairs, the robots need to rely on their individual beliefs, which may diverge.

Assume there are three robots in the group. One robot believes that the room is occupied and thus, according to it, the group should not adopt the goal. According to the other two robots, the group should adopt the goal. One robot believes that the garbage bin is full and the floors clean and the other that the floors are dirty. The question is how to should one aggregate the beliefs of the robots in this case? The majority of the robots would estimate that the goal should be adopted. However the group is not univocal
and as a result, the goal would not be chosen for a group goal when the robots reason according to \[9, 12, 16\]. A general method for forming group attitudes needs to specify how group attitudes are formed also when agents have disagreeing beliefs or limited persuasive abilities. A framework that allows for such general methods is still lacking.

Consider now that the group adopts the goal, but before pursuing it, the robots learn that there is a seminar scheduled in the meeting room. The group has to de-commit from their intention to clean the room. However, to be able to do so, the group has to have a commitment strategy that allows de-committing upon change in the reasons for the goal adoption. Furthermore, the group needs to be able to reconsider its reasons for goals, and goals, after de-committing. If they simply drop the goal, without “remembering” why, they would not be able to re-deliberate and commit again once the seminar is over.

An intelligent agent reacts to the changes in its environment and a group should be capable of doing the same. When new information becomes available the group faces a choice: to remain committed to its group attitudes or to reconsider them. The most sensitive commitment strategy proposed by Rao and Georgeff \[21\] allows for a group reaction only when the goal is accomplished or impossible. However, the existence of a goal is intertwined with the existence of some beliefs \[2\]. Consequently, there is a need for a commitment strategy that reacts to new information regarding those beliefs on which the goal hinges. Furthermore, groups need to know not only when to de-commit from their goals but also how to reconsider their goal-related beliefs.

The research question we address in this paper is:

**How can groups choose and reconsider their goals?**

A good methodology for collectively choosing and reconsidering goals is one that can be used by any group of agents regardless of the homogeneity of the individual beliefs of its members and their persuasion abilities. The relation between individual goals and beliefs can be specified and analyzed in modal agent logics like $BDI_{LTL}$ \[22\]. The challenge in group goal generation is to incorporate the aggregation of individual attitudes into collective ones, as studied in merging, judgment aggregation and social choice \[3, 11, 13, 14, 17\]. However, using this approach is not straightforward. The main difficulty lies in the inability to use judgment aggregation directly in a $BDI_{LTL}$ framework. Properties of judgment aggregation and modal agent logic are of different kinds, as they were initially developed for different purposes.

Our research question thus breaks down into the following sub-questions:

1. *How to aggregate individual (epistemic and motivational) attitudes?*
2. *What are the desirable properties for this aggregation?*

A good methodology for collectively choosing and reconsidering goals is also a methodology that is dynamic enough to allow for the group to change its epistemic and motivational attitudes in light of new information. Having such a methodology increases the autonomy and reactivity of groups. Hence we need to answer the following sub-question as well:

3. *Which commitment and reconsideration strategies should be available for groups?*
We thus focus on finding the following solutions:

**Formal framework.** We extend a multi-agent modal language to be able to represent judgment aggregation in it.

**Choice of aggregation.** Judgment aggregation is an abstract framework that allows for various desirable properties to be specified for the aggregation procedure. The task is to determine which aggregation properties are necessary and desirable for aggregating individual beliefs and goals.

**Commitment and reconsideration strategies.** Within our formal framework, we define when to de-commit from intentions and how to change them.

**Multiple goals.** Since a group can have more than one goal, we need to model the effect that the commitment to (and reconsideration of) one goal has over the commitment to (and reconsideration of) other goals.

We make the following assumptions. The group has a fixed membership – agents do not join or depart from the group. The group has a set of candidate group goals and an order of priority over these goals. The decision rules for each candidate goal are available to the group. How the decision rules are learned is outside of the scope of this paper. Here we do not consider how plans are generated, executed and revised once the group goals are selected or reconsidered, nor we consider whether the group goals are executed via individual or joint actions.

The layout of the paper is as follows. In Section 2 we introduce judgment aggregation. In Section 3 we extend a multi-modal agent formalism to capture the aggregation of individual attitudes. Sections 4 and 5 respectively study the commitment and reconsideration strategies. Related work, conclusions and outlines for future work are in Section 6.

## 2 From individual attitudes to group goals

Let us consider again the example of the robot cleaning crew from Section 1.

**Example 1.** Let $C = \{w_1, w_2, w_3\}$ be a crew of cleaning robots. We denote the group goal to clean the meeting room with $g_1$, and the reasons to adopt this goal with: there are no people in the room ($p_1$), the room is dirty ($p_2$), the garbage bin in it is full ($p_3$).

The individual beliefs of the robots on whether $g_1$ should be the group goal are justified by individual beliefs on $p_1, p_2, p_3$ using the decision rule $\left( p_1 \land (p_2 \lor p_3) \right) \leftrightarrow g_1$.

A group of agents could collectively decide to adopt or reject a goal by voting. However, the goals of the agents are not independent from their beliefs [2], which we express using decision rules. When the decision is whether to adopt a goal or not, we also need to explain why this goal should (not) be adopted. Having reasons for (not) adopting a goal enables the agents to reconsider this goal in light of new information. Judgment aggregation deals with the problem of reaching decisions for a set of logically dependent issues, by aggregating individual opinions on these issues.

### 2.1 Judgment aggregation preliminaries

A general overview of judgment aggregation is given in [17]. Here we present the terminology and definitions of judgment aggregation we use in our framework.
Consider a logic $\mathcal{L}$ with entailment operator $\models$. In a judgment aggregation framework, an agenda $\mathcal{A} \subseteq \mathcal{L}$ is the pre-defined set of issues, on which every agent casts her judgments. E.g., the agenda of Example 1 is $\mathcal{A} = \{p_1, p_2, p_3, g_1\}$. Consider a valuation function $\nu$ such that a judgment “yes” on issue $a$ is a valuation $\nu(a) = 1$, while a judgment “no” on the same issue is a valuation $\nu(a) = 0$. A profile is the set of all judgments assigned, on the agenda issues, by the decision-making agents.

**Definition 1 (Profile).** Let $\mathcal{N} = \{1, 2, \ldots\}$ be a set of agent names, $\mathcal{A} \subseteq \mathcal{L}$ an agenda and $\overline{\mathcal{A}} = \mathcal{A} \cup \{\neg a \mid a \in \mathcal{A}\}$. A profile $\pi$ is the set of judgments for the agenda items, submitted by the agents in the group: $\pi \subseteq \mathcal{N} \times \overline{\mathcal{A}}$. We define two operators:

- The judgment set for agent $i$ is $\pi \Rightarrow i = \{a \mid (i, a) \in \pi\}$.
- The set of all the agents who judged “yes” to $a \in \overline{\mathcal{A}}$ is $\pi \Downarrow a = \{i \mid (i, a) \in \pi\}$.

Definition 1 extends the definition of a profile given [17] with the operators $\Rightarrow$ and $\Downarrow$. We introduce these operators to ease the explanation of the various aggregation properties we present later on. To get a better intuitive grasp on these operators, the reader should envision the profile as a two-dimensional object with the agenda items identifying the columns and the agents identifying the rows:

$$
\begin{array}{cccc}
p_1 & p_2 & p_3 & g_1 \\
w_1 & 1 & 1 & 0 & 1 \\
w_2 & 0 & 1 & 1 & 1 \\
w_3 & 1 & 0 & 0 & 0 \\
\end{array}
$$

$\pi$ is a possible profile for Example 1. We identify $\pi \Rightarrow w_2$ as the row labeled $w_2$ and $\pi \Downarrow p_2$ as the 1 entries in the column labeled $p_2$, which identify the agents who casted judgement “yes” on $p_2$.

In judgment aggregation, the judgments over $\mathcal{A}$, both individual and aggregates, are constrained by decision rules $\mathcal{R} \subseteq \mathcal{L}$. The set $\mathcal{R}$ contains only formulas $r$ such that all the non-logical symbols of $\mathcal{A}$ occur in $r \in \mathcal{R}$. For instance, in Example 1, the decision rule is: $(p_1 \land (p_2 \lor p_3)) \leftrightarrow g_1$ and all agents respect it. In general, each agent could follow a different decision rule and yet another decision rule can be imposed for the group. In judgment aggregation, the agents are allowed to submit only those judgment sets which are consistent with $r \in \mathcal{R}$. Often, the agents are also required to cast judgments on all the agenda issues. We construct Definition 2 to formalize what it means for a judgment set to be admissible.

**Definition 2 (Admissible profiles).** A judgment set $\pi \Rightarrow i$ is complete if and only if for every $a \in \mathcal{A}$, either $(i, a) \in \pi$ or $(i, \neg a) \in \pi$. A judgment set $\pi \Rightarrow i$ is consistent with $r \in \mathcal{R}$ if and only if $\{r\} \cup (\pi \Rightarrow i) \neq \bot$. A judgment set $\pi \Rightarrow i$ is admissible if it is consistent with the given $r \in \mathcal{R}$ and complete for $\mathcal{A}$. The set of all admissible judgment sets for a given $\mathcal{A}$ and $r$ is denoted by $\mathcal{W}$.

A profile $\pi$ is admissible if $\pi \Rightarrow i$ is admissible for all $i \in \mathcal{N}$. The set of all admissible profiles for agents $\mathcal{N}$ is denoted by $\Pi$.

We denote $\text{con}(r, \varphi) = 1$ if $\varphi$ is consistent with $r$ and $\text{con}(r, \varphi) = 0$ otherwise. For the profile in Figure 1, $\text{con}((p_1 \land (p_2 \lor p_3)) \leftrightarrow g_1, \pi \Rightarrow i) = 1$ for every $i \in C$.

We now present the definition of a judgment aggregation function, as given in [17].
**Definition 3 (Judgment aggregation function).** Given an agenda $A$ and agents $N$, a judgment aggregation (JA) function is $f : 2^N \times A \to 2^A$.

We refer to $f(\pi)$ as the collective judgment set for $\pi$. We denote the result of $f(\pi)$ with $\bot$ when the JA function produces an inadmissible judgment set. If $f(\pi) \neq \bot$ then we call $f(\pi)$ a decision and denote it by $D_{\pi}$. Figure 1 illustrates an example of a judgment aggregation function and profile, for which the collective judgment set is $\bot$ because $con((p_1 \land (p_2 \lor p_3)) \leftrightarrow g_1, \{p_1, p_2, p_3, \neg g_1\}) = 0$.

The JA function is a useful abstraction, because many properties of judgment aggregation can be defined in terms of it. It then can be studied which properties can be accepted together (avoiding impossibility results). Given a JA function $f$, we describe the most common properties in the JA literature.

**Universal domain.** $f$ satisfies universal domain if and only if its domain is $\Pi$.

**Anonymity.** Given a profile $\pi \in \Pi$, let $\pi = \{\pi \in 1, \ldots, \pi = n\}$, be the multiset of all the individual judgment sets in $\pi$. Two profiles $\pi, \pi' \in \Pi$ are permutations of each other if and only if $p_{\pi} = p_{\pi'}$. $f$ satisfies anonymity if and only if $f(\pi) = f(\pi')$ for all permutation $\pi$ and $\pi'$.

**Unanimity on $a \in A$.** $f$ satisfies unanimity on $a \in A$ if and only if for every profile $\pi \in \Pi$, it holds: if for all $i \in N$, $(i, a) \in \pi$, then $a \in f(\pi)$.

**Collective rationality.** $f$ satisfies collective rationality if and only if for all $\pi \in \Pi$, $con(\pi, f(\pi)) = 1$ for a given $r \in R$, and either $a \in f(\pi)$ or $\neg a \in f(\pi)$ for every $a \in A$.

**Constant.** $f$ is constant when there exists $\varphi \in 2^A$ such that for every $\pi \in \Pi$, $f(\pi) = \varphi$.

**Independence.** Let $\Phi = \{\pi \downarrow a \mid a \in A\}$ for every $\pi \in \Pi$. Let $f_1, \ldots, f_m$ be functions defined as $f_j : A \times \Phi \to \{0, 1\}$. The JA function $f$ satisfies independence if and only if for all $\pi \in \Pi$, there exist functions $f_j$ such that for all $\varphi \in f(\pi)$ it holds that $\varphi = \{a \mid a \in A, f_j(a, \pi \downarrow a) = 1\} \cup \{-a \mid a \in A, f_j(a, \pi \downarrow a) = 0\}$.

The best known example of $f_j : A \times \Phi \to \{0, 1\}$ is the simple majority voting $f_m$ which counts how many agents expressed judgment “yes” on agenda item $a$. The function $f_m$ returns $a$ if that number of agents is greater than $\lceil \frac{n}{2} \rceil$ and $\neg a$ otherwise.

![Fig. 1. A profile, issue-wise majority aggregation, premise-based and conclusion-based majority.](image-url)
The JA function $f_{maj}$ satisfies universal domain, anonymity, unanimity, completeness, and independence but it does not satisfy collective rationality, as it can be seen on Figure 1. All judgement aggregation functions which satisfy universal domain, anonymity, independence and collective rationality are constant [17]. The most debated [3,17] is independence. The reason why it is convenient to have independence is because it is a necessary condition to guarantee the non-manipulability of $f$ [8]. An aggregation function is non-manipulable, if no agent can obtain his sincere judgment set $\phi$ selected as the collective judgment set by submitting another judgment set $\phi'$.

Two aggregation procedures that violate independence but guarantee universal domain, anonymity and collective rationality have been proposed in the literature: premise-based and conclusion-based procedures. A distinguishing feature of judgment aggregation with respect to social choice theory is the distinction between premises and conclusions. The agenda is a union of two disjoint sets: the premise set ($\mathcal{A}^p$), and the conclusion set ($\mathcal{A}^c$). $\mathcal{A} = \mathcal{A}^p \cup \mathcal{A}^c$, $\mathcal{A}^p \cap \mathcal{A}^c = \emptyset$. In Example 1, $\mathcal{A}^p = \{p_1,p_2,p_3\}$ and $\mathcal{A}^c = \{q_2\}$. We give a general definition on when a JA function is premise- or conclusion-based in Definition 4. The literature on judgment aggregation [17] defines premise- and conclusion-based procedures only in terms of issue-wise majority.

**Definition 4.** Let $\pi^p = \{(i,a) \mid (i,a) \in \pi, a \in \mathcal{A}^p\}$, $\pi^c = \{(i,a) \mid (i,a) \in \pi, a \in \mathcal{A}^c\}$ and let the premise and conclusion aggregation functions be defined as $f^p : \Pi \mapsto 2^{\mathcal{A}^p}/\emptyset$ and $f^c : \Pi \mapsto 2^{\mathcal{A}^c}/\emptyset$ correspondingly. The JA function $f$ is premise-based if and only if there exists a $f^p$ such that $f^p(\pi^p) \subseteq f(\pi)$ and conclusion-based if and only if there exists a $f^c$ such that $f^c(\pi^c) \subseteq f(\pi)$, for all $\pi \in \Pi$.

An example of premise and conclusion-based aggregations is given in Figure 1. Intuitively, a JA function is premise-based when the collective judgment set on the premises is obtained as a result of aggregating only the premise profile $\pi^p$. The decisions $f(\pi)$ are those $\phi \in \mathcal{W}$ for which $f^p(\pi^p) \subset \phi$ (or alternatively $f^c(\pi^c) \subset \phi$ for conclusion-based aggregations). However, it is possible that, depending on the decision rule, there are more than one $\phi \in \mathcal{W}$ that satisfy the condition $f^p(\pi^p) \subset \phi$ (or alternatively $f^c(\pi^c) \subset \phi$). This is what happens for the conclusion-based procedure we illustrate in Figure 1.

### 2.2 Judgment aggregation for BDI agents

In this section first we set the problem of finding collective decisions for group goals in the context of judgment aggregation. We then argue that the aggregation function used for this problem should satisfy: collective rationality, universal domain, unanimity and select a unique $\phi \in \mathcal{W}$. For a democratic group, in which all agents have equal say on what the group attitudes should be, anonymity should be satisfied. For a group in which the agents have different levels of expertise, anonymity can be omitted.

We use $Gg$ to denote that “$g$ is a group goal”. A set of the candidate group goals is the set $\mathcal{G} = \{Gg \mid g \in \mathcal{L}\}$ which contains all the goals which the group considers to adopt. For each goal $Gg \in \mathcal{G}$, the group has at its disposal decision rules $\mathcal{R}_g \subseteq \mathcal{R}$ and an agenda $\mathcal{A}_g$ composed of the goal $g$ (conclusion), and all the reasons (premises) relevant for $g$, which were given by the decision rule. Each agent submits her judgments on the agenda thus generating a profile $\pi_g$, such that $\text{con}((\mathcal{R}_g, \pi_g \Rightarrow i)) = 1$ for all $i \in \mathcal{N}$.
The decision rules contain all the constraints which the individual and collective judgment sets should satisfy. These constraints contain three types of information: rules describing how the goal depends on the reasons (justification rules $R^\text{just}_g$), rules describing the constraints of the world inhabited by the agents (domain knowledge $R^\text{DK}_g$) and rules that describe how $g$ interacts with other candidate goals of the group (coordination rules $R^\text{coord}_g$). Hence, the decision rule for a group goal $g$ is $R_g = R^\text{just}_g \cup R^\text{DK}_g \cup R^\text{coord}_g$.

Example 2 (Example 1 revisited). Consider the cleaning crew from Example 1. $R^\text{just}_{g_1}$ is $(p_1 \land (p_2 \lor p_3)) \Rightarrow G_{g_1}$ and $A_{g_1} = \{p_1, p_2, p_3, G_{g_1}\}$. Suppose that the crew has the following candidate group goals as well: place the furniture in its designated location ($g_2$) and collect recyclables from garbage bin ($g_3$). The agendas are $A_{g_2} = \{p_4, p_5, p_6, p_7, G_{g_2}\}$, $A_{g_3} = \{p_8, p_9, p_3, G_{g_3}\}$. The justification rules are $R^\text{just}_{g_2} = (p_4 \land p_5 \land (p_6 \lor p_7)) \Rightarrow G_{g_2}$ and $R^\text{just}_{g_3} = (p_8 \land p_9 \land p_3) \Rightarrow G_{g_3}$. The formulas $p_4 \lor p_5$ are: the furniture is out of place ($p_4$), the designated location for the furniture is empty ($p_5$), the furniture has wheels ($p_6$), the furniture has handles ($p_7$), the agents can get revenue for recyclables ($p_8$), there is a container for the recyclables ($p_9$).

An example of a domain knowledge could be $R^\text{DK}_g = \neg p_4 \Rightarrow \neg p_5$, since it can not happen that the designated location for the furniture is empty while the furniture is not out of place. Group goal $G_{g_3}$ can be pursued at the same time as $G_{g_1}$, however, $G_{g_2}$ can only be pursued alone. Thus the coordination rule for all three goals is $R^\text{coord}_{g_1} = R^\text{coord}_{g_2} = R^\text{coord}_{g_3} = ((G_{g_2} \land \neg (G_{g_1} \lor G_{g_3})) \lor \neg G_{g_2})$.

We want the justifications for a goal to explain when a goal should be adopted/refuted. Having collective justifications for a goal enables the agents to, when the world changes, consider adopting a goal that has been rejected previously. To this end, we take into consideration justification rules which are of form $G_g \leftrightarrow \Gamma$, where $\Gamma \in \mathcal{L}$ such that all the non-logical symbols of $\Gamma$ occur in $A^c_g$ and $\{G_g\} = A^c_g$.

We now continue to discuss the desirable properties for the aggregation of individual beliefs and goals. We need a JA function that satisfies universal domain to be able to aggregate all admissible profiles. We can use only JA functions that satisfy collective rationality. If the collective set is not complete, for example, if it contains only a collective judgment on the goal, then we do not know why the goal was (not) adopted and consequently we would not know when to revise it. For example, the cleaning crew decides for the goal $g_3$ (collect recyclables), without having the reasons like $p_9$ (a container where to put them). If the world changes and $\neg p_9$ holds, the robots will continue to collect recyclables.

The aggregation of all admissible profiles should produce a set consistent with the decision rule because otherwise we would not be generating the group goals and justifications for them. For the same reason we need an aggregation method that selects a unique $\varphi \in \mathcal{W}$.

To guarantee that $\text{con}(R_g, f(\pi_g)) = 1$, we need to choose between conclusion-based and premise-based procedures. At first glance, the premise-based procedure seems an obvious choice since it will produce complete collective judgment sets under our decision rules. However, upon closer inspection, this procedure has notable drawbacks.

As it is observable from the profile in Figure 1; a premise-based procedure may lead
the group to adopt a conclusion that the majority (or even the unanimity) of the group does not endorse. In our case, the conclusion is the goal and a premise-based aggregation may establish a group goal which none of the agents is interested in pursuing. To avoid this we need to aggregate using a conclusion-based procedure. In particular we want a conclusion-based procedure that has the property of unanimity on $G_g$.

Given that our decision rules are of the form $g \leftrightarrow \Gamma$, there exist profiles for which a conclusion-based procedure will not produce complete collective set of judgments. However, the conclusion-based aggregation can be supplemented with an additional procedure that completes the produced set of judgments when necessary. Such aggregation procedure is the complete conclusion-based procedure (CCBP) developed in [20]. This CCBP satisfies universal domain and is collectively rational. However, it does not always produce a unique decision. CCBP produces a unique collective judgment for the goal, but it can generate more than one set of justifications for it. This is an undesirable, but not an unmanageable property. To deal with ties, as it is the practice with voting procedures, the group can either determine a default set of justifications for adopting/rejecting each candidate goal, or it can appoint one member of the group as tiebreaker. Tie-breaking problems in judgment aggregation are the focus of our ongoing research.

The CCBP from [20] also satisfies anonymity. Whether this is a desirable property for a group of artificial agents depends entirely on whether the group is democratic or the opinions of some agents are more important. CCBP can be adjusted to take into consideration different weights on different agents’ judgment sets.

3 Formal framework for modeling group attitudes

In this section we introduce the language of modal multi-agent logic in which we represent individual and collective mental attitudes. We then combine the methodology of judgment aggregation with this representation language and show how collective attitudes are generated. To model commitment to a group goal and reconsideration of a group goal we use temporal logic.

3.1 Modal multi-agent logic

Just like modal agent logic is concerned mainly with the relation between the individual goals and beliefs over time, modal multi-agent logic is concerned with the relation between group goals and beliefs over time. We use modal multi-agent logic to represent: the agenda, individual judgments, collective judgments and new information that may prompt goal revision. In line with judgment aggregation proper, we do not use the formal language to represent the judgment aggregation function, but only the arguments of this function and the results from it. We assume that there is a service, available to the agents, that elicits the individual judgments, performs the aggregation and makes the aggregation results available, to the members and for plan-generation.

Agenda issues in judgment aggregation are usually represented by propositional formulas. This is not a viable option in our case. First, we want to represent the difference between a goal and the supporting reasons by means of representing the distinction between conclusions and premises explicitly in the logic. Second, the logic should represent the distinction between individual and collective judgments. We distinguish
conclusions from premises by using a single $K$ modal operator $Gg$ for representing that “g is a group goal”.

The obvious choice for modeling the judgment “true” on agenda issue $a$, of an agent $i$, is the modal operator belief $B_{i,a}$ (correspondingly $B_{i,¬a}$ for a judgment “false” on $a$). However, we find that beliefs are ill suited for modeling collective judgments of agents. While a belief $B_{i,a}$ is an exclusively private mental state, judgments are public and contributed for the decision-making purposes of the group. A judgment is thus closer to a public commitment than to a private belief. Hence, we model judgments by using the modal operator of acceptance $A_{S,a}$ [19]. $A_{S,a}$ denotes: agents in $S$ accept $a$. The operator $A_{S,a}$ allows us to represent both individual judgments, $S = \{i\}$, for $i \in \mathcal{N}$ and collective judgments with $S = \mathcal{N}$. We define the group acceptance $A_{\mathcal{N},a}$ to be the result of applying judgment aggregation to the individual acceptances. We present the formal definitions of profile and judgment aggregation function in our logic in Definition 6. The modal operator $A_{S}$ we use is inspired by the modal operator of the acceptance logic of [19]. The details on the relation between acceptance logic and our acceptance operator are given in the Appendix.

We represent the new information that becomes available to the agents with a normal modal $K$ operator $E.a$ denotes: “it is observed that $a$ is true”.

Lastly, to model how the group attitudes evolve with reconsideration we need a temporal logic. By using $LTL$ we do not need to distinguish between path formulas and state formulas, but we are able to quantify over traces. Just as in $BDI_{LTL}$, where $B\Box a$ denotes that $a$ is believed to be necessary, we use $E\square a$ to denote that $a$ is observed to be impossible.

We now give the syntax of our modal multi-agent logic $AGELTL$.

**Definition 5 (Syntax).** Let $Agt$ be a non-empty set of agents, with $S \subseteq Agt$, and $L_P$ be a set of atomic propositions. The admissible formulae of $AGELTL$ are formulae $\psi_0, \psi_1$ and $\psi_2$ of languages $L_{prop}$, $L_G$ and $L_{AGELTL}$ correspondingly:

\[
\psi_0 ::= p \mid (\psi_0 \land \psi_0) \mid \neg \psi_0
\]

\[
\psi_1 ::= \psi_0 \mid G\psi_0
\]

\[
\psi_2 ::= \psi_0 \mid A_{S}\psi_1 \mid E\psi_2 \mid X\psi_2 \mid (\psi_2 U \psi_2)
\]

where $p$ ranges over $L_P$ and $S$ over $2^{Ag}$. Moreover, $\Diamond \phi \equiv \top U \phi$, $\Box \phi \equiv \neg \Diamond \neg \phi$, and $\phi R \phi' \equiv \neg (\phi U \neg \phi')$.

We can now adjust the definition for a judgment aggregation function. We represent individual judgments with $A_{i,a}$ with intuitive reading “agent $i$ judges $a$ as true” and $A_{i,\neg a}$ with reading “agent $i$ judges $a$ as false”.

**Definition 6 (JA in AGELTL).** Let $\mathcal{N} = \{1, 2, \ldots\}$ be a set of agent names and $\mathcal{G} \subseteq L_G / L_{prop}$ a set of candidate goals.

An agenda $A_g \subseteq L_G$ for goal $Gg \in \mathcal{G}$ is a set of formulas such that $A_g = \overline{A}_g \cup \overline{A}_g^c$, with $\overline{A}_g = A_g^p \cup \{\neg a \mid a \in A_g^p\}$, $A_g^p \subseteq L_{prop}$ and $\overline{A}_g = \{Gg, \neg Gg\}$.

A profile of judgments is the set $\pi = \{A_{i,a} \mid i \in \mathcal{N}, a \in A_g\}$.

$\pi \Rightarrow i = \{a \mid A_{i,a} \in \pi\}$ is the judgment set of agent $i$.

$\pi \Downarrow a = \{i \mid A_{i,a} \in \pi\}$ is the set of all the agents that accepted $a$.

Given a set of decision rules $\mathcal{R}_g \subseteq L_{prop}$, a profile is acceptable if and only if, for all $i \in \mathcal{N}$ and all $a \in \mathcal{A}$, $\text{con}(\mathcal{R}_g, \pi \Rightarrow i) = 1$ and either $A_{i,a}$ or $A_{i,\neg a}$. The set of all
acceptable profiles for $N$ and $A_p$, given $\mathcal{R}_g$ is $\mathcal{P}$.

The decision for a profile $\pi$, $D_{\pi} = \{a \in f^a(\pi), f^a : \mathcal{P} \mapsto 2^A\}$.

For instance, the profile in Figure 1, is $\pi_{g_1} = \{A_{(w_1)}p_1, \neg A_{(w_2)}p_2, \neg A_{(w_3)}p_3, A_{(w_4)}\neg G_{g_1}, A_{(w_5)}\neg p_1, \ldots, A_{(w_9)}G_{g_1}\}$. The decision using premise-based majority would be $\{A_{C}p_1, A_{C}p_2, A_{C}p_3, A_{C}G_{g_1}\}$.

$AGE_{LTL}$ can be used to model that an agent does not have a judgment on an agenda issue, but we will work with the assumption that either $A_{(i)}$ or $A_{(i)}\neg a$ for all agents and agenda issues.

$AGE_{LTL}$ has Kripke semantics. As in Schild [22], a Kripke structure is defined as a tuple $M = \langle W, R, g, E, A, L \rangle$. $W$ is a set of possible situations, and $R$ is the temporal relation over situations $r \subseteq W \times W$. $g$ is the goal relation over situations $g \subseteq W \times W$, while $E$ is the observation relation over situations $e \subseteq W \times W$. Let $\Delta = 2^W \times Inst$. $A : \Delta \mapsto W \times W$ maps every $s \in \Delta$ to a relation $A_s$ between possible situations. $L$ is a truth assignment to the primitive propositions of $L_P$ for each situation $w \in W$, i.e., $L(w) : Prop \mapsto \{true, false\}$.

Temporal formulas are validated in the standard manner [15]. Normal modal formulas $G\psi$ and $E\psi$ have standard semantics, see for example [4]. Acceptance formulas $AS\psi$ are validated according to the semantics of acceptance logic presented in [19]. The axiomatization of $AGE_{LTL}$ is given in the Appendix. Note that $AGE_{LTL}$ is a fusion of the decidable logics: $LTL$, acceptance logics and two $K$ modal logics.

3.2 Generation of group goals

The mental state of the group is determined by the mental states of the members and the choice of judgment aggregation function. We represent the mental state of the group by a set $\chi$ of $AGE_{LTL}$ formulas. The set $\chi$ contains: (a) the set of all candidate goals for the group $g \subseteq L_G/L_{prop}$ and, for each $G_g \in g$, the corresponding decision rules $\mathcal{R}_g$, as well as the individual and collective acceptances made in the group regarding agenda $A_g$. The set $\chi$ is common knowledge for the group members. An agent uses $\chi$ when it acts as a group member and its own beliefs and goals when it acts as an individual.

To deal with multiple, possibly mutually inconsistent goals, the group has a priority order $\succeq_x$ over the group goals $g \subset \chi$. To avoid overburdening the language with a $\succeq_x$ operator, we incorporate the priority order within the decision rules $\mathcal{R}_g^{just}$,We want the decision rules to capture that if $G_{g_1}$ is not consistent (according to the coordination rules) with some higher priority goals $G_{g_1}, \ldots, G_{g_m}$, then the group can accept $G_{g_j}$ if and only if none of $G_{g_1}, \ldots, G_{g_m}$ is accepted. Hence, we replace the justification rule $\mathcal{R}_g^{just}$ with $\mathcal{R}_g^{just} \equiv (I_1 \wedge \bigwedge_j^m (A_{N \neg G_{g_j}}) \leftrightarrow G_{g_1}$, where $G_{g_j} \in g, G_{g_j} \succeq_x G_{g_i}$ and $G_{g_j} \wedge G_{g_j} \wedge \mathcal{R}_{g_j}^{coord} \models \bot$.

Example 3. Consider the goals and rules of the robot crew $C$ from Example 2. Assume the crew has been given the priority order $G_{g_1} \succ_g G_{g_2} \succ_g G_{g_3}$. $\chi$ contains: $g = \{G_{g_1}, G_{g_2}, G_{g_3}\}$, one background knowledge rule, one coordination rule, three justification rules, out of which two are new priority modified rules:

$\{G_{g_1} \neg p_4 \leftrightarrow \neg p_5, (G_{g_2} \land \neg (G_{g_1} \lor G_{g_3})) \lor \neg G_{g_2}, G_{g_1} \leftrightarrow (p_1 \land (p_2 \lor p_3)) \}$.

$G_{g_2} \leftrightarrow \neg (p_4 \land p_5 \land (p_6 \lor p_7) \land A_{C \neg G_{g_1}}), G_{g_3} \leftrightarrow (p_8 \land p_9 \land \neg (A_{C \neg G_{g_2}}))$. 

124
The agents give their judgments on one agenda after another starting with the agenda for the highest priority candidate goal. Once the profile $\pi$ and the decision $D\pi$ for a goal $g$ are obtained, they are added to $\chi$. To avoid the situation in which the group casts judgments on an issue that has already been decided, we need to remove decided issues from $A_g$ before eliciting the profile for this agenda.

The group goals are generated by executing $\text{GenerateGoals}(\chi, N)$.

function $\text{GenerateGoals}(\chi, S)$:
for each $G_g \in S$ s.t. $[\forall G_j \in S : (G_j \succeq G_j) \Rightarrow (A_g \neg G_j \in \chi \lor A_g \neg G_j \in \chi)]$

$\{ B := \{a \mid A_g \neg a \in \chi \} \cup \{ \neg a \mid A_g \neg a \in \chi \}\} \cap A_g;$

comment if $B$ is the set of already collectively accepted issues from $A_g$,

$A_g^* := A_g / B;$

$\pi_g := elicit(S, A_g^*, \chi);$:

$\chi := \chi \cup \pi_g \cup f^a(\pi_g);$:

return $\chi.$

$elicit$ requests the agents to submit complete judgment sets for $\pi_g \subset \chi$. We require that $elicit$ is such that for all returned $\pi$ it holds $con(\chi, \pi \Rightarrow i) = 1$ for all $i \in N$ and that $con(\chi, f^a(\pi)) = 1$. When a higher priority goal $G_g$ is accepted by the group, a lower priority incompatible goal $G_j$ cannot be adopted regardless of the judgments on the issues in $A_j$. Nevertheless, although $elicit$ will provide individual judgments for the agenda $A_g$, if the acceptance of $G_g$ is reconsidered, we can obtain a new decision on $G_j$ because the profile for $G_j$ is available.

**Example 4.** Consider the $\chi$ for robots given in Example 3. The following calls to $elicit$ are made in the given order. First, $\pi_{g_1} = elicit(N, A_{g_1}^*, \chi)$ with the $\text{GenerateGoals}(\chi) = \chi' = \chi \cup \pi_{g_1} \cup f^a(\pi_{g_1})$. Second, $\pi_{g_2} = elicit(N, A_{g_2}^*, \chi')$, with $\text{GenerateGoals}(\chi') = \chi'' = \chi' \cup \pi_{g_2} \cup f^a(\pi_{g_2})$. Last, $\pi_{g_3} = elicit(N, A_{g_3}^*, \chi'')$, with $\text{GenerateGoals}(\chi'') = \chi''' = \chi'' \cup \pi_{g_3} \cup f^a(\pi_{g_3})$. Since there is no overlapping between agendas $A_{g_1}$ and $A_{g_1}$, $A_{g_2}^* \equiv A_{g_1}$ and $A_{g_3}^* \equiv A_{g_2}$. However, since $A_{g_2} \cap A_{g_3} = p_3$, then $A_{g_3}^* = \{p_3, p_3, G_{g_3}\}$.

**Proposition 1.** $\text{GenerateGoals}$ terminates if and only if $elicit$ terminates and does not violate the candidate goal preference order.

The proof is straightforward.

### 4 Commitment strategies

The group can choose to reconsider the decision (acceptance or rejection) on a group goal in presence of new information. Whether the group chooses to reconsider depends on how committed it is to bring the goal about. According to Cohen and Levesque [5], an agent intends $g$ if it has chosen to pursue goal $g$ and it committed itself to making $g$ happen. Following the same line of reasoning, we define group intention to be $I_N g \equiv A_N G g$ and read it as “the agents $N$ intend $g$”. We defined collective acceptance as resulting from judgment aggregation, which is a social choice method. Thus, in our framework, group intention is social choice with commitment. The level of commitment of a group to a collective acceptance depends on the choice of commitment strategy.
These are the three main commitment strategies (introduced by Rao and Georgeff [21]):

Blind commitment: $I_i g \rightarrow (I_i g \cup B_i g)$

Single-minded commitment: $I_i g \rightarrow (I_i g \cup (B_i g \lor B_i \neg g))$

Open-minded commitment: $I_i g \rightarrow (I_i g \cup (B_i g \lor \neg G_i g))$

These commitment strategies only consider the relation between beliefs regarding $g$ and $Gg$. Instead, a commitment to a goal can now be reconsidered upon new information on either one of the agenda issues in $A_g$ and also upon new information on a higher priority goal.

The strength of our framework is exhibited in its ability to describe the groups’ commitment not only to its decision to adopt a goal, but also to its decision to reject a goal. Namely, if the agents decided $I_N g_i$ and $A_N \neg g_j$, are committed to both $I_N g_i$ and $A_N \neg g_j$. Commitment to reject $g$ allows for $g$ to be reconsidered and eventually adopted if the state of the world changes.

Let $N$ be a set of agents with a set of candidate goals $G$. Let $Gg_i, Gg_j \in G$ have agendas $A_{g_i}, \ A_{g_j}$. We use $p \in A^p_{g_i}$ and $q_i \in A^c_{g_i}, \ q_j \in A^c_{g_j}$. The profiles and decisions are $\pi_i$, and $f^\pi(\pi_i)$; $Gg_j > Gg_i$, and $Gg_j$ cannot be pursued at the same time as $Gg_i$.

We use the formulas $(\alpha_1) - (\alpha_7)$ to refine the blind, single-minded and open-minded commitment. Instead of the "until", we use the temporal operator release: $\psi R \phi \equiv \neg (\neg \psi U \neg \phi)$, meaning that $\phi$ has to be true until and including the point where $\psi$ first becomes true; if $\psi$ never becomes true, $\phi$ must remain true forever. Unlike the until operator, the release operator does not guarantee that right hand-side formula will ever become true, which in our case translates to the fact that an agent could be forever committed to a goal.

\[
\begin{align*}
(\alpha_1) & \quad E_{g_i} R I_N g_i \\
(\alpha_2) & \quad \bot R A_N \neg G_{g_i} \\
(\alpha_3) & \quad (E \neg g_i \lor E_{g_i}) R A_N q_i \\
(\alpha_4) & \quad A_N \neg q_j R A_N q_i \\
(\alpha_5) & \quad A_N p \rightarrow (E \neg p R A_N q_i)
\end{align*}
\]

**Blind commitment:** $\alpha_1 \land \alpha_2$.

Only the observation that the goal is achieved ($E_{g_i}$) can release the intention to achieve the goal $I_N g_i$. If the goal is never achieved, the group will always be committed to it. If a goal is not accepted, then the agents will not reconsider accepting it.

**Single-minded commitment:** $\alpha_3$.

Only new information on the goal (either that the goal is achieved or had become impossible) can release the decision of the group to adopt / reject the goal. Hence, new information is only regarded if it concerns the conclusion, while information on the remaining agenda items is ignored.

**Extended single-minded commitment:** $\alpha_3 \land \alpha_4$.

Not only new information on $g_i$, but also the collective acceptance to adopt a more important incompatible goal $g_j$ can release the intention of the group to achieve $g_i$. Similarly, if $g_i$ is not accepted, the non-acceptance can be revised, not only if $g_j$ is observed to be impossible or achieved, but also when the commitment to pursue $g_j$ is dropped (for whatever reason).
Open-minded commitment: $\alpha_3 \land \alpha_5$.
A group will maintain its collective acceptances to adopt/reject a goal as long as the new information regarding all collectively accepted agenda items is consistent with $f^a(\pi_g)$.

Extended open-minded commitment: $\alpha_3 \land \alpha_4 \land \alpha_5$.
Extending on the single-minded commitment, a change in intention to pursue a higher priority goal $G_{g_j}$ can also release the acceptance of the group on $G_{g_j}$.
Once an intention is dropped, a group may need to reconsider its collective acceptances. This may cause for the dropped goal to be re-affirmed, but a reconsideration process will be invoked nevertheless.

5 Reconsideration strategies

In Section 3.2 we defined the mental state of the group $\chi$. We can now define what it means for a group to be coherent.

Definition 7 (Group coherence). Given a Kripke structure $M$ and situations $s \in W$, a group of $N$ agents is coherent if the following conditions are met:

$(\rho_1):$ $M, s \models \neg(\bigwedge_{a \in A_g} \neg a)$ for any $S \subseteq N$ and any $a \in A_g$.
$(\rho_2):$ If $M, s \models \chi$ then $\chi \not\perp$.
$(\rho_3):$ $M, s \models \bigwedge G \rightarrow \neg \Box \neg g$ for all $G \in G$.
$(\rho_4):$ Let $Gg \in G$ and $G' = G' \setminus \{Gg\}$, then $M \models \bigwedge G \land \bigwedge E \neg g \rightarrow X(\neg Gg)$.
$(\rho_5):$ Let $p \in A_p^g$ and $q \in \{Gg, \neg Gg\}$, $Ep \land (Ep \land p \land (Ep \land Aq^g) \rightarrow XA \land p$.

The first condition ensures that no contradictory judgments are given. The second condition ensures that the mental state of the group is logically consistent in all situations. The third and fourth conditions ensure that impossible goals cannot be part of the set of candidate goals and if $g$ is observed to be impossible in situation $s$, then it will be removed from $G$ in the next situation. $\rho_5$ enforces the acceptance of the new information on the group level, when the commitment strategy so allows – after $a$ is observed and that lead the group to de-commit from $g$, the group necessarily accepts $a$.

A coherent group accepts the observed new information on a premise. This may cause the collective acceptances to be inconsistent with the justification rules. Consequently, the decisions and/or the profiles in $\chi$ need to be changed in to ensure that $\rho_1$ and $\rho_2$ are satisfied. If, however $\Box \neg g$ or $g$ is observed, the group reconsider $\chi$ by removing $Gg$ from $G$. In this case, the decisions and profiles are not changed.

For simplicity, at present we work with a world in which the agents’ knowledge can only increase, namely the observed information is not a fluent. A few more conditions need to be added to the definition of group coherence, for our model to be able to be applicable to fluents. E.g., we need to define which observation is accepted when two subsequent contradictory observations happen.

For the group to be coherent at all situations, the acceptances regarding the group goals need to be reconsidered after de-commitment. Let $D_g \subset \chi$ contain the group acceptances for a goal $g$, while $\pi_g \subset \chi$ contain the profile for $g$. There are two basic ways in which a collective judgment set can be reconsidered. The first way is to elicit a new profile for $g$ and apply judgment aggregation to it to obtain the reconsidered $D_g^*$. The second is to reconsider only $D_g$ without re-eliciting individual judgments. The first approach requires communication among agents. The second approach can be done.
by each agent reconsidering $\chi$ by herself. We identify three reconsideration strategies available to the agents. The strategies are ordered from the least to the most demanding in terms of agent communication.

**Decision reconsideration ($D$-$r$).** Assume that $E p, p \in A_g^p, q \in \{Gg, \neg Gg\}$ and the group de-commits from $A_N^g q$. The reconsidered decision $D_g^q$ is such that $p$ is accepted, i.e., $A_N^g p \in D_g^q$, and the entire decision is consistent with the justification rules, namely $\text{con}(R_q^{just}, D_g^q \Rightarrow N) = 1$. If the $D$-$r$ specifies an unique $D_g^q$, for any observed information and any $D_g$, then $\chi$ can be reconsidered without any communication among the agents. Given the form of $R_q^{just}$ (see Section 3.2), this will always be the case.

However $D$-$r$ is not always an option when the de-commitment occurred due to a change in collective acceptance of a higher priority goal $g'$. Let $q' \in \{Gg', \neg Gg'\}$. Let the new acceptance be $A_N^g q'$. $D$-$r$ is possible if and only if $D_g^q = D_g$ and $\text{con}(R_g^{just}, D_g \cup \{A_N^g \neg q'\}) = 1$. Recall that $A_N^g q'$ was not in $A_g$ and as such the acceptance of $q'$ or $\neg q'$ is never in the decision for $\pi_g$.

**Partial reconsideration of the profile (Partial $\pi$-$r$).** Assume that $E a, a \in A_g, Gg \in G$. Not only the group, but also the individual agents need to accept $a$. The Partial $\pi$-$r$ asks for new individual judgments be elicited. This is done to ensure the logical consistency of the individual judgment sets with the observations. New judgments are only elicited from the agents $i$ which $A_{\{i\}}^g a$.

Let $W \subseteq N$ be the subset of agents $i$ s.t. $A_{\{i\}}^g a \in \chi$. Agents $i$ are s.t. $A_{\{i\}}^g a \in \chi$ when the observation is $E a$. Let $\pi_g^W \subseteq \pi_g$ be the set of all acceptances made by agents in $W$. We construct $\chi' = \chi/\pi_g^W$. The new profile and decision are obtained by executing $\text{GenerateGoals}(\chi', W)$.

**Example 5.** Consider Example 4. For $\pi_{g_1}, \pi_{g_2}$ and $\pi_{g_3}$ of the robot crew C, the decisions $D_{g_1} = \{A_CP_1, A_C \neg P_2, A_CP_3, A_C Gg_1\}, D_{g_2} = \{A_CP_4, A_CP_5, A_CP_6, A_CP_7, A_C \neg Gg_2\}$ and $D_{g_3} = \{A_CP_8, A_CP_9, A_C Gg_3\}$ are made. Assume the group de-commits on $Gg_1$ because of $E \neg P_2$. If the group is committed to $Gg_3$, the commitment on $Gg_3$ will not allow for $A_N^g P_3$ to be modified when reconsidering $Gg_1$. Since $A_N^g P_3$ exists in $\chi'$, $P_3$ will be excluded from the (new) agenda for $g_1$, although it was originally in it. elicit calls only on the agents in $W$ to complete $\pi_{g_1} \in \chi'$ with their judgment sets.

**Full profile reconsideration ($\pi$-$r$).** The full profile reconsideration is the same with the partial reconsiderations in all respects except one – now $W = N$. Namely, within the full profile revision strategy, each agent is asked to revise his judgment set by accepting the new information, regardless whether he had already accepted the information or not.

### 5.1 Combining revision and commitment strategies

Unlike the Rao and Georgeff commitment strategies [21], in our framework the commitment strategies are not axioms of the logic. We require that the commitment strategy is valid in all the models of the group and not in all the models of $AGE_{LTL}$. This allows the group to define different commitment strategies and different revision strategies for different goals. It might even choose to revise differently depending on which information triggered the revision. Choosing different revision strategies for each goal, or each
type of new information, should not undermine the coherence of the group record \( \chi \). The conditions of group coherence of the group ensures that after every reconsideration \( \chi \) must remain consistent. However, some combinations of commitment strategies can lead to incoherence of \( \chi \).

**Example 6.** Consider the profiles and decisions in Example 5. Assume that initially the group chose open-minded commitment for \( I_{Cg1} \) and blind commitment for \( I_{Cg3} \), with goal open-minded commitment for \( A_{Cg2} \). If \( E_{g1} \) and thus \( I_{Cg1} \) is dropped, then the extended open-minded commitment would allow \( A_{Cg2} \) to be reconsidered and eventually \( I_{Cg2} \) established. However, since the group is blindly committed to \( I_{Cg3} \), this change will not cause reconsideration and as a result both \( I_{Cg2} \) and \( I_{Cg3} \) will be in \( \chi \) thus making \( \chi \) incoherent.

Problems arise when \( \text{sub}(R^{just}_g) \cap \text{sub}(R^{just}_j) \neq \emptyset \), where \( \text{sub}(R^{just}_g) \) denotes the set of atomic sub-formulas of \( g \) \( (G_{g1}, G_{g3} \in \mathcal{G}) \). Proposition 2 summarizes under which conditions these problems are avoided.

**Proposition 2.** Let \( \alpha' \) and \( \alpha'' \) be the commitment strategies selected for \( g_i \) and \( g_j \) correspondingly, \( \chi \cap \alpha' \cap \alpha'' \neq \bot \) (in all situations):

a) if \( \phi \in \text{sub}(R^{just}_g) \cap \text{sub}(R^{just}_j) \) and \( p \in \mathcal{A}_{g_i} \cap \mathcal{A}_{g_j} \), then \( \alpha_5 \) is either in both \( \alpha' \) and \( \alpha'' \) or in none;

b) if \( G_{g_i} \) is more important than \( G_{g_j} \) and \( G_j \) and \( G_i \) cannot be accepted at the same time, then \( \alpha_4 \in \alpha'' \).

**Proof.** The proof is straightforward. Namely, if the change on acceptance of \( G_{g_i} \) causes the decision on \( G_{g_j} \) to induce group incoherence, we are able to de-comit from \( G_{g_j} \). If we were not able to de-comit on \( G_{g_j} \) group coherence is blocked. If the change on collective acceptance on \( G_{g_j} \) is caused by an observation on a premise \( p \in \mathcal{A}_{g_i} \cap \mathcal{A}_{g_j} \), then condition a) ensures that the commitment to the collective acceptance regarding \( G_{g_j} \) does not block group coherence. If the change on collective acceptance on \( G_{g_j} \) is caused by a change in commitment to a higher priority goal the condition b) ensures that a commitment regarding \( G_{g_j} \) does not block group coherence. Condition b) allows only “goal sensitive” commitments to be selected for lower level goals.

### 6 Conclusions

We constructed a group decision-making framework by combining judgment aggregation and multi-agent modal logic. We identified the desirable judgment aggregation properties for aggregation in collaborative groups. Our multi-agent modal logic \( AGE_{LTL} \) is an extension of \( BDI_{LTL} \) with modal operators for representing individual and collective acceptances and observations of new information. We extend the commitment strategies of Rao and Georgeff [21] to increase the reactivity of the group to new information. Having a group goal \( G_{g} \) in our framework does not imply that the members individually have the goal \( G_{g} \) and groups can have different levels of commitment to different goals.

Our framework is intended for groups that engage in joint activity. Our framework is applicable when it is not possible to assume that the agents persuade each other on a single position and goal, but it is necessary anyway that the group presents itself as
a single whole from the point of view of beliefs and goals, and above all as a rational entity that has goals justified by the beliefs it holds, and it is able to revise these goals under the light of new information. This requirement was held by Tuomela [24] and adopted in agent theory by Boella and van der Torre [1] and Lorini [18]. There are many situations where the proposal of the paper can be applied. For example in an open-source project, where several people have to discuss online to agree on which is their position on issues (e.g. which algorithm is better for a certain task) and which is their goal (e.g. delivering a new realize on which date).

Work on collaborative group activities [9, 12, 16] and group decision-making protocols [10] focus on how to define collective intentionality and how to distribute the collective intentions over the agents. We define group intention to be the collective acceptance of a group goal and focus on defining commitment strategies for the collective acceptances. An advantage of our framework is its ability to allow groups to commit to a decision to reject a goal, thus having the option to reconsider rejected goals. Furthermore, we do not only show when to reconsider, but also how, by defining reconsideration strategies. Table 1 summarizes our commitment and reconsideration strategies.

<table>
<thead>
<tr>
<th>Commitment to</th>
<th>Release on</th>
<th>Change</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A (\neg) G g$</td>
<td>$\square \neg g$</td>
<td>$G g_j$</td>
<td>$A_j^g$</td>
</tr>
<tr>
<td>Blind</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Single-minded</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extended</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Open-minded</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extended</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1. $G g_j \succ G g$ and can not be pursued at the same time with $G g, \otimes D (g)$ denotes: collective attitudes for $g$ are reconsidered. $\otimes \pi_g$ denotes: the profile (all or some parts of it) is re-elicited.

Icard et al. [23] consider the joint revision of individual attitudes, with the revision of beliefs triggering intention revision. However, they do not allow for the revision of intentions to cause a revision of beliefs. We focus on joint reconsideration of group attitudes and we allow for both the change in epistemic and in motivational attitudes can be a cause for reconsideration.

In our framework, the new information is simultaneously available to the entire group. In the future we intend to explore the case when only some members of the group observe the new information. The only assumptions we make regarding the connectivity of the members is that they are able to communicate their acceptances and receive the aggregation result. The problem of elicitation and communication complexity in voting is a nontrivial one [6, 7] and we intend to study these properties within our framework.

In the work we presented, we do not consider how individual acceptances are formed. We can take that $B_i \phi \rightarrow A_{\{i\}} \phi$, but this need not be the case. We can define dishonest agents as those for which $B_i \phi \rightarrow A_{\{i\}} \phi$ does not hold. In this case, the agent might declare $A_{\{i\}} \phi$ while it does not believe $\phi$. The question is whether there are scenarios in which incentives for doing so arise. Furthermore, given that the some reconsideration strategies call for re-elicitation of judgments, can an agent have the incentive to deliberately give judgments that would lead to sooner re-elicitation? We intend to devote more attention to answering these questions as well as studying the manipulability properties of our decision-making framework.
References

Appendix – Relations between $AGE_{LTL}$ and acceptance logic and axiomatization of $AGE_{LTL}$

Here we elaborate in more detail on the fusion logic $AGE_{LTL}$ we use. The modal operator $A_S \phi$ we use is equivalent to the modal operator $A_{S,x} \phi$ of the acceptance logic of [19] with one syntactic and one semantic exception.

The operator $A_{S,x} \phi$ uses $x$ ranging over a set of labels to describe the context under which the acceptance is made. In our case the context is that of the group and since we deal with only one group, we have no use of these labels. The context labels play no role in the semantics of the acceptance logic formulas.

On the semantic level, the axioms for $A_S \phi$ are all the axioms of $A_{S,x} \phi$ except two: the axiom inclusion ($Inc.$) and the axiom unanimity ($Un.$). Dropping ($Un.$) does not affect the decidability of the logic of acceptance. ($Inc.$) (not to be confused with unanimity in judgment aggregation) states that if $A_{N,x} \phi$, then $\forall i \in N, A_{\{i\},x} \phi$. In our case, it is the aggregation of individual acceptances that determines the collective acceptance. Since we use the acceptances to model judgments, we do not want an axiom that states that the individual judgments mirror the collective judgments. The agents use the collective acceptance when functioning as a group and their private beliefs when acting as individuals. In our framework we do not model the private mental states, but only individual acceptances which are “declared” to all the agents in the group.

($Inc.$) states that if a the group $C$ accepts $\varphi$, so will any subgroup $B \subseteq C$. In our case, the judgment aggregation over the profile containing only the judgment sets of $B$ can produce a different collective judgment set than the profile containing all the judgment sets of $C$.

The axiomatization of the $AGE_{LTL}$ logic is thus:

(ProTau) All principles of propositional calculus

(LTL Tau) All axioms and derivation rules of LTL

(K-G) $G(\phi \rightarrow \psi) \rightarrow (G\phi \rightarrow G\psi)$

(K-E) $E(\phi \rightarrow \psi) \rightarrow (E\phi \rightarrow E\psi)$

(K-A) $A_S(\phi \rightarrow \psi) \rightarrow (A_S\phi \rightarrow A_S\psi)$

(PAccess) $A_S\phi \rightarrow A_M A_S\phi$ if $M \subseteq S$

(NAccess) $\neg A_S\phi \rightarrow A_M \neg A_S\phi$ if $M \subseteq S$

(Mon) $\neg A_S \bot \rightarrow \neg A_M \bot$ if $M \subseteq S$

(MP) From $\vdash \phi$ and $\vdash (\phi \rightarrow \psi)$ infer $\vdash \psi$

(Nec-A) From $\vdash \phi$ infer $\vdash A_S \phi$

(Nec-G) From $\vdash \phi$ infer $\vdash G\phi$

(Nec-E) From $\vdash \phi$ infer $\vdash E\phi$

Given $\mathcal{M} = \langle W, \mathcal{R}, \mathcal{G}, \mathcal{E}, \mathcal{A}, L \rangle$ and $s \in W$, the truth conditions for the formulas of $AGE_{LTL}$ (in a situation $s$) are:

- $\mathcal{M}, s \not\models \bot$;
- $\mathcal{M}, s \models p$ if and only if $p \in L(p)$;
- $\mathcal{M}, s \models \neg \phi$ if and only if $\mathcal{M}, s \not\models \phi$;
- $\mathcal{M}, s \models \phi \land \psi$ if and only if $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models A_{N} \phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in \mathcal{A}$;
- $\mathcal{M}, s \models G\phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in G$;
- $\mathcal{M}, s \models E\phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in E$;
- $\mathcal{M}, s \models X\phi$ if and only if $\mathcal{M}, s' \models \phi$ for some $(s', (s, s')) \in R$;
- $\mathcal{M}, s \models \phi U\psi$ if and only if $\mathcal{M}, s \models \phi$; $\mathcal{M}, s' \models \phi$ for all $s', i \in \{1, 2, \ldots, k\}$ such that $\{(s, s^1), (s^1, s^2), \ldots, (s^{k-1}, s^k)\} \in R$ and $(s^{k+1}, s^{k+1}) \in R$; it holds $\mathcal{M}, s^{k+1} \not\models \phi$ and $\mathcal{M}, s^{k+1} \models \psi$.

A formula $\phi$ is true in an $AGE^{LTL}$ model $\mathcal{M}$ if and only if $\mathcal{M}, s \models \phi$ for every situation $s \in W$. The formula $\phi$ is valid (noted $\models_{AGE^{LTL}}$) if and only if $\phi$ is true in all $AGE^{LTL}$ models. The formula $\phi$ is $AGE^{LTL}$-satisfiable if and only if the formula $\neg \phi$ is not $AGE^{LTL}$ valid.
MERCURIO: An Interaction-oriented Framework for Designing, Verifying and Programming Multi-Agent Systems*

Matteo Baldoni¹, Cristina Baroglio¹, Federico Bergenti⁴, Antonio Boccalatte³, Elisa Marengo¹, Maurizio Martelli³, Viviana Mascardi³, Luca Padovani¹, Viviana Patti¹, Alessandro Ricci², Gianfranco Rossi³, and Andrea Santi²

¹ Università degli Studi di Torino
{baldoni,baroglio,emarengo,padovani,patti}@di.unito.it
² Università degli Studi di Bologna
{a.ricci,a.santi}@unibo.it
³ Università degli Studi di Genova
{martelli,mascardi}@disi.unige.it, nino@disi.unige.it
⁴ Università degli Studi di Parma
{federico.bergenti,gianfranco.rossi}@unipr.it

Abstract. This is a position paper reporting the motivations, the starting point and the guidelines that characterize the MERCURIO project proposal, submitted to MIUR PRIN 2009. The aim is to develop formal models of interactions and of the related support infrastructures, that overcome the limits of the current approaches by explicitly representing not only the agents but also the computational environment in terms of rules, conventions, resources, tools, and services that are functional to the coordination and cooperation of the agents. The models will enable the verification of interaction properties of MAS from the global point of view of the system as well as from the point of view of the single agents, due to the introduction of a novel social semantic of interaction based on commitments and on an explicit account of the regulative rules.

1 Motivation

The growing pervasiveness of computer networks and of Internet is an important catalyst pushing towards the realization of business-to-business and cross-business solutions. Interaction and coordination, central issues to any distributed system, acquire in this context a special relevance since they allow the involved groups to integrate by interacting according to the agreed contracts, to share best practices and agreements, to cooperatively exploit resources and to facilitate the identification and the development of new products.

---

* Position paper
⁵ Italian name of Hermes, the messenger of the gods in Greek mythology.
The issues of interaction, coordination and communication have been receiving great attention in the area of Multi-Agent Systems (MAS). MAS are, therefore, the tools that could better meet these needs by offering the proper abstractions. Particularly relevant in the outlined application context are a shared and inspectable specification of the rules of the MAS and the verification of global properties of the interaction, like the interoperability of the given roles, as well as properties like the conformance of an agent specification (or of its run-time behavior) to a protocol. In open environments, in fact, it is important to have guaranties on how interaction will take place, coping with notions like responsibility and commitment. Unfortunately, current proposals of platforms and languages for the development of MAS do not supply high level tools for directly implementing this kind of specifications. As a consequence, they do not support the necessary forms of verification, with a negative impact on the applicability of MAS to the realization of business-to-business and cross-business systems.

Let us consider, for instance, JADE [4, 18, 16, 17], which is one of the best known infrastructures, sticking out for its wide adoption also in business contexts. JADE agents communicate by exchanging messages that conform to FIPA ACL [3]. According to FIPA ACL mentalistic approach, the semantics of messages is given in terms of preconditions and effects on the mental states of the involved agents, which are assumed to share a common ontology. Agent platforms based on FIPA exclusively provide syntactic checks of message structures, entrusting the semantics issues to agent developers. This hinders the applicability to open contexts, where it is necessary to coordinate autonomous and heterogeneous agents and it is not possible to assume mutual trust among them. In these contexts it is necessary to have an unambiguous semantics allowing the verification of interaction properties before the interaction takes place [52] or during the interaction [9], preserving at the same time the privacy of the implemented policies.

The mentalistic approach does not allow to satisfy all these needs [40]; it is suitable for reasoning from the local point of view of a single agent, but it does not allow the verification of interaction properties of a MAS from a global point of view. One of the reasons is that the reference model lacks an abstraction for the representation, by means of a public specification, of elements like (i) resources and services that are available in the environment/context in which agents interact and (ii) the rules and protocols, defining the interaction of agents through the environment/context. All these elements belong to (and contribute to make) the environment of the interacting agents. Such an abstraction, if available, would be the natural means for encapsulating resources, services, and functionalities (like ontological mediators) that can support the communication and the coordination of agents [67, 66, 43], thus facilitating the verification of the properties [13]. It could also facilitate the interaction of agents implemented in different languages because it would be sufficient that each language implements the primitives for interacting with the environment [1]. One of the consequences of the lack of an explicit representation of the environment is that only forms of direct commu-
communication are possible. On the contrary, in the area of distributed systems and also in MAS alternative communication models, such as the generative communication based on tuple spaces [32], have been put forward. These forms of communication, which do not necessarily require a space-time coupling between agents, are not supported.

The issues that we mean to face have correspondences with issues concerning normative MAS [70] and Artificial Institutions [31, 65]. The current proposals in this field, however, do not supply all of the solutions that we need: either they do not account for indirect forms of communication or they lack mechanisms for allowing the a priori verification of global properties of the interaction. As [31, 65] witness, there is, instead, an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts. In this case, institutional actions are performed by executing instrumental actions that are conventionally associated with them. Currently, instrumental actions are limited to speech acts but this representation is not always natural. For instance, for voting in the human world, people often raise their hands rather than saying the name corresponding to their choice. If the environment were represented explicitly it would be possible to use a wider range of instrumental actions, that can be perceived by the other agents through the environment that acts as a medium.

Our goal is, therefore, to propose an infrastructure that overcomes such limits. The key of the proposal is the adoption of a social approach to communication [45, 14, 13, 12], based on a model that includes an explicit representation not only of agents but also of their environment, as a collection of virtual and physical resources, tools and services, “artifacts” as intended in the Agents & Artifacts (A&A) meta-model [43], which are shared, used and adapted by the agents, according to their goals. The introduction of environments is fundamental to the adoption of an observational (social) semantics, like the one used in commitment protocols, in that it supplies primitives that allow agents to perceive and to modify the environment itself and, therefore, to interact and to coordinate with one another in a way that satisfies the rules of the environment. On the other hand, the observational semantics is the only sufficiently general semantics to allow forms of interaction and of communication that do not rely solely on direct speech acts. As a consequence we will include models where communication is mediated by an environment, that encapsulates and applies rules and constraints aimed at coordinating agents at the organization level, and integrates ontological mediation functionalities. The environment will provide the contract that agents should respect and a context into which interpreting their actions. In this way, it will be possible to formally verify the desired properties of the interaction, a priori and at execution time.

2 Vision

The focus of our proposal is on the definition of formal models of interactions and of the related support infrastructures, which explicitly represent not only the agents but also the environment in terms of rules of interaction, conventions,
resources, tools, and services that are functional to the coordination and cooperation of the agents. These models must allow both direct and indirect forms of communication, include ontological mediators, and enable the verification of interaction properties of MAS from the global point of view of the system as well as from the point of view of the single agents. The approach we plan to pursue in order to define a formal model of interaction is based on a revision in social terms of the interaction and of the protocols controlling it, along the lines of [14, 13, 12]. Furthermore, we will model the environment, in the sense introduced by the A&A meta-model [43]. This will lead to the study of communication forms mediated by the environment. The resulting models will be validated by the implementation of software tools and of programming languages featuring the designed abstractions. More in details, with reference to Fig. 1, the goals are:

**Fig. 1. The MERCurIO architecture.**

**To introduce a formal model** for specifying and controlling the interaction. The model (top level of Fig. 1) must be equipped with an observational (commitment-based) semantics and must be able to express not only direct communicative acts but also interactions mediated by the environment. This will enable forms of verification that encompass both global interaction properties and specific agent properties such as interoperability and conformance [11]. The approach does not hinder agent autonomy, it guarantees the privacy of the policies implemented by the agents, and consequently favors the composition of heterogeneous agents. The model will be inspired by the social approach introduced in [45] and subsequently extended in [14, 13, 12].

**To define high-level environment models** supporting the forms of interactions and coordination between agents outlined above. These models must
support: interaction protocols based on commitments; the definition of rules on the interaction; forms of mediated communication and coordination between agents (such as stigmergic coordination). They must also enable forms of a priori and runtime verification of the interaction. To these aims, we plan to use the A&A meta-model [58, 67, 43, 56] and the corresponding notion of programmable environment [57] (programming abstractions level of Fig. 1).

**To integrate ontologies and ontological mediators** in the definition of the models so as to guarantee openness and heterogeneity of MAS. Mediation will occur at two distinct levels: the one related to the vocabulary and domain of discourse and the one that characterizes the social approach where it is required to bind the semantics of the agent actions with their meaning in social terms. Ontological mediators will be realized as artifacts.

**To integrate the abstractions** defined in the above models within programming languages and frameworks. In particular, we plan to integrate the notions of agents, of environment, of direct and mediated communication, and of ontological mediators. Possible starting points are the aforementioned FIPA ACL standard and the works that focus on the integration of agent-oriented programming languages with environments [55]. The JaCa platform [57], integrating Jason and CArtAgO, will be taken as reference. This will form the execution platform of Fig. 1 and will supply the primitives for interacting with the environments.

**To develop an open-source prototype** of software infrastructure for the experimentation of the defined models. The prototype will integrate and extend existing technologies such as JADE [18, 16, 17] (as a FIPA-compliant framework), CArtAgO [1] (for the programming and the execution of environments), Jason (as a programming language for BDI agents), MOISE [35] (as organizational infrastructure).

**To identify applicative scenarios** for the evaluation of the developed models and prototypes. In this respect we regard the domain of Web services as particularly relevant because of the need to deploy complex interactions having those characteristics of flexibility that agents are able to guarantee. Another interesting application regards the verification of adherence of bureaucratic procedures of public administration with respect to the current normative. Specific case studies will be defined in collaboration with those companies that have stated interest towards the project.

### 3 State of Art

These novel elements, related to the formation of and the interaction within decentralized structures, find an initial support in proposals from the literature in the area of MAS. Current proposals, however, are still incomplete in that they supply solutions to single aspects. For instance, electronic institutions [28, 10, 35, 34] regulate interaction, tackle open environments and their semantics allows the verification of properties but they only tackle direct communication protocols, based on speech acts, and do not include an explicit notion of environment. Commitment protocols [45, 69], effective in open systems and allowing more general
forms of communication, do not supply behavioral patterns, and for this reason it is impossible to verify properties of the interaction. Eventually, most of the models and architectures for environments prefigure simple/reactive agent models without defining semantics, that are comparable to the ones for ACL, and without explaining how such proposals could be integrated with direct communication models based on speech acts. We classify the relevant contributions in the literature according to the objectives and the methodological aspects that will be examined in-depth along the project.

3.1 Formal Models for Regulating the Interaction in MAS

This topic has principally been tackled by modeling interaction protocols. Most of protocol representations refer to classic models, such as Petri nets, finite state machines, process algebras, and aim at capturing the expected interaction flow. An advantage of this approach is that it supports the verification of interaction properties [52, 21, 11], such as: verifying the interoperability of the system and verifying if certain modifications of a system preserve some desired properties (a crucial issue in open domains where agents can enter/leave the system at any time). Singh and colleagues criticize the use of procedural specifications because too rigid [60, 24, 69]: agents cannot take advantage of opportunities that emerge along the interaction and that are not foreseen by their procedure. Another issue is that communication languages use a BDI semantics (FIPA ACL is an example), where each agent has goals and beliefs of its own. At the system level, however, it is impossible to perform introspection of agents, which are, for this reason, black boxes. For what concerns the verification of properties this approach allows agents to draw conclusions about their own behavior but not to verify global properties of the system [40, 64].

Both problems are solved by commitment protocols [45, 60], which rely on an observational semantics of the interaction and offer adequate flexibility to agents. Moreover, they do not require the spatio-temporal coupling of agents (as instead direct communication does). Another advantage is that, though remaining black boxes, agents agree on the meaning of the social actions of the protocol. Since interactions are observable and their semantics is shared, each agent should be able to draw conclusions concerning the system as a whole. Unfortunately, besides some preliminary studies [61], the state of art does not contain proposals on how performing the verifications in a MAS, ruled by this kind of protocols. A relevant feature seems to be the introduction, within commitment protocols, of behavioral rules which constrain the possible evolutions of the social state [13, 12].

3.2 Environment Models

The notion of environment has always played a key role in the context of MAS; recently, it started to be considered as a first-class abstraction useful for the design and the engineering of MAS [67]. A&A [43] follows this perspective, being a meta-model rooted upon Activity Theory and Computer Support Cooperative
Work that defines the main abstractions for modeling a MAS, and in particular for modeling the environment in which a MAS is situated. A&A promotes a vision of an endogenous environment, that is a sort of software/computational environment, part of the MAS, that encapsulates the set of tools and resources useful/required by agents during the execution of their activities. A&A introduces the notion of artifact as the fundamental abstraction used for modeling the resources and the tools that populates the MAS environment. The introduction of the environment as a new first-class abstraction requires new engineering approaches for programming the MAS environment. The CArtAgO framework [57] has been devised precisely for coping with this new necessity. It provides the basis for the engineering of MAS environments on the base of: (i) a proper computational model and (ii) a programming model for the design and the development of the environments on the base of the A&A meta-model. In particular, it provides those features that are important from a software engineering point of view: abstraction, it preserves the agent abstraction level, since the main concepts used to define application environments, i.e. artifacts and workspaces, are first-class entities in the agents world, and the interaction with agents is built around the agent-based concepts of action and perception (use and observation); modularity and encapsulation, it provides an explicit way to modularize the environment, where artifacts are components representing units of functionality, encapsulating a partially-observable state and operations; extensibility and adaptation, it provides a direct support for environment extensibility and adaptation, since artifacts can be dynamically constructed (instantiated), disposed, replaced, and adapted by agents; reusability, it promotes the definition of types of artifact that can be reused as tools in different application contexts, such as in the case of coordination artifacts empowering agent interaction and coordination, such as blackboards and synchronizers. These features will be advantageous in the realization of the second goal of the project, w.r.t. approaches like [25], where commitment stores, communication constraints and the interaction mechanisms reside in the middleware, which shields them from the agents. This has two disadvantages: the first is that even though all these elements are accounted for in the high level specification, the lack of a corresponding programming abstraction makes it difficult to verify whether the system corresponds to the specification; the second is a lack of flexibility, in that it is not possible for the agents to dynamically change the rules of interaction or to adopt kinds of communication that are not already implemented in the middleware.

In the state of the art numerous applications of the endogenous environments, i.e. environments used as a computational support for the agents’ activities, have been explored, including coordination artifacts [44], artifacts used for realizing argumentation by means of proper coordination mechanisms [42], artifacts used for realizing stigmergic coordination mechanisms [54, 48], organizational artifacts [34, 49, 50]. Even if CArtAgO can be considered a framework sufficiently mature for the concrete developing of software/computational MAS environments it can not be considered “complete” yet. Indeed at this moment the state of the art and in particular the CArtAgO framework are still lacking: (i) a reference standard
on the environment side comparable to the existing standards in the context of
the agents direct communications (FIPA ACL), (ii) the definition of a rigorous
and formal semantics, in particular related to the artifact abstraction, (iii) an
integration with the current communication approaches (FIPA ACL, KQML,
etc.), and finally (iv) the support of semantic models and ontologies.

3.3 Multi-agent Organizations and Institutions

The possibility of controlling and specifying interactions is relevant also for areas
like the organizational theory [39, 70, 15, 35] and electronic institutions [28, 10]
areas. Tendentiously, the focus is orthogonal to the one posed on interaction
protocols, in that it concerns the modeling of the structure rather than of the
interaction.

The abstract architecture of e-Institutions (e.g. Ameli [28]), places a middle-
ware composed of governors and staff agents between participating agents and
an agent communication infrastructure (e.g. JADE [18, 16, 17]). The notion of
environment is dialogical: it is not something agents can sense and act upon but
a conceptual one that agents, playing within the institution, can interact with by
means of norms and laws, based on specific ontologies, social structures, and lan-
guage conventions. Agents communicate with each other by means of speech acts
and, behind the scene, the middleware mediates such communication. The ex-
tension proposed for situated e-Institutions [10] introduces the notion of “World
of Interest” to model the environment, that is external to the MAS but which is
relevant to the MAS application. The infrastructure of the e-Institution, in this
case, mediates also the interaction of the agents in the MAS with the view of
the environment that it supplies. Further along this line, but in the context of
organizations, ORA4MAS [34] proposes the use of artifacts to enable the access
of the agents in the MAS to the organization, providing a working environment
that agents can perceive, act upon and adapt. Following the A&A perspective,
they are concrete bricks used to structure the agents’ world: part of this world is
represented by the organizational infrastructure, part by artifacts introduced by
specific MAS applications, including entities/services belonging to the external
environment.

According to [10] there are, however, two significant differences among ar-
tifacts and e-Institutions: (i) e-Institutions are tailored to a particular, though
large, family of applications while artifacts are more generic; (ii) e-Institutions
are a well established and proven technology that includes a formal foundation,
and advanced engineering and tool support, while for artifacts, these features are
still in a preliminary phase. One of the aims of MERCURIO is to give to artifacts
both the formal foundation (in terms of commitments and interaction patterns)
and the engineering tools that they are still missing. The introduction of inter-
action patterns with an observational nature, allowing the verification of global
properties, that we aim at studying, will allow the realization of e-Institutions
by means of artifacts. The artifact will contain all the features necessary for
monitoring the on-going interactions and for detecting violations. A second step
will be to consider organizations and realize them again by means of artifacts.
To this aim, it is possible to exploit open source systems like CArtAgO [1], for the programming and the execution of environments, and MOISE [35], as organizational infrastructure.

3.4 Semantic Mediation in MAS

The problem of semantic mediation at the vocabulary and domain of discourse levels was faced for the first time by the “Ontology Service Specification” [8] issued by FIPA in 2001. According to that specification, an “Ontology Agent” (OA, for short) should be integrated in the MAS in order to provide services such as translating expressions between different ontologies and/or different content languages and answering queries about relationships between terms or between ontologies. Although the FIPA Ontology Service Specification represents the first and only attempt to analyze in a systematic way the services that an OA should provide for ensuring semantic interoperability in an open MAS, it has many limitations. The main one is the assumption that each ontology integrated in the MAS adheres to the OKBC model [6]. Currently, in fact, the most widely accepted ontology language is OWL [7] which is quite different from OKBC and cannot be converted to it in an easy and automatic way. Also, agents are allowed to specify only one ontology as reference vocabulary for a given message, which is a strong limitation since an agent might use terms from different ontologies in the same message, and hence it might want to refer to more than one ontology at the same time.

Maybe due to these limitations, there have been really few attempts to design and implement OAs. The first dates back to 2001 [62] and realizes an OA for the COMTEC platform that implements a subset of the services of a generic FIPA-compliant OA. In 2007 [46] integrated an OA into AgentService, a FIPA compliant framework based on .NET [63]. Ontologies in AgentService are represented in OKBC, and hence the implementation of their OA is fully compliant with the FIPA specification, although the offered services are a subset of the possible ones. The only two attempts of integrating a FIPA-compliant OA into JADE, we are aware of, are [41], and [23]. Both follow the FIPA specification but adapt it to ontologies represented in OWL. The first proposal is aimed at storing and modifying OWL ontologies: the OA agent exploits the Jena library [36] to this aim. The second proposal, instead, faces the problem of “answering queries about relationships between terms or between ontologies”. The solution proposed by the authors exploits ontology matching techniques [29]. Apart from [23], no other existing proposal faces that problem. Among non-FIPA-compliant solutions, we mention [37], which focuses on the process of mapping and integrating ontologies in a MAS thanks to a set of agents which collaborate together, and the proposal in [47], which implements the OA as a web service, in order to offer its services also over the Internet.

As far as semantic mediation at the social approach level is concerned, we are aware of no proposals in the literature. In order to take the context of count-as rules into account, we plan to face this research issue by exploiting context aware semantic matching techniques, that extend and improve those described in [38].
3.5 Software Infrastructures for Agents

The tools currently available to agent developers fail in supporting both semantic interoperability and goal-directed reasoning. Nowadays, the development of agents and multi-agent systems is based on two kinds of tools: agent platforms and BDI (or variations) development environments. Agent platforms, such as JADE [18, 16, 17] and FIPA-OS [2] provide only a transport layer and some basic services, but they do not provide any support for goal-directed behavior. Moreover, they lack support for semantic interoperability because they do not take into account the semantics of the ACL they adopt. The available BDI development environments, such as Jadex [22] and 2APL [27], support only syntactic interoperability because they do not exploit their reasoning engines to integrate the semantics of the adopted ACL.

The research on Agent Communication Languages (ACL) is constantly headed towards semantic interoperability [33] because the most common ACLs, e.g., KQML [30] and FIPA ACL [3], provide each message with a declarative semantics that was explicitly designed to support goal-directed reasoning. Unfortunately, the research on ACLs only marginally investigated the decoupling properties of this kind of languages (see, e.g., [19, 20]). To support the practical development of software agents, several programming languages have thus been introduced to incorporate some of the concepts from agent logics. Some languages use actions as their starting point to define commitments (Agent-0, [59]), intentions (AgentSpeak(L), [53]) and goals (3APL, [26]).

4 Expected Results

The achievements expected from this research are of different natures: scientific result that will advance the state of the art, software products deriving from the development of implementations, and upshots in applicative settings.

The formal model developed in MERCURIO will extend commitment protocols by introducing behavioral rules. The starting point will be the work done in [14, 13, 12]. This will advance the current state of the art with respect to the specification of commitment protocols and also with respect to the verification of interaction properties (like interoperability and conformance), for which there currently exist only preliminary proposals [61]. Another advancement concerns the declarative specification of protocols and their usage by designers and software engineers. The proposals coming from MERCURIO conjugate the flexibility and openness features that are typical of MAS with the needs of modularity and compositionality that are typical of design and development methodologies. The adoption of commitment protocols makes it easier and more natural to represent (inter)actions that are not limited to communicative acts but that include interactions mediated by the environment, namely actions upon the environment and the detection of variations of the environment ruled by “contracts”.

For what concerns the coordination infrastructure, a first result will be the definition of environments based on the A&A meta-model and on the CArtAgO
computational framework, that implement the formal models and the interaction protocols mentioned above. A large number of the environments, described in the literature supporting communication and coordination, have been stated considering purely reactive architectures. In MERCURIO we will formulate environment models that allow goal/task-oriented agents (those that integrate pro-activities and re-activities) the participation to MAS. Among the specific results related to this, we foresee an advancement of the state of the art with respect to the definition and the exploitation of forms of stigmergic coordination [54] in the context of intelligent agent systems. A further contribution regards the flexible use of artifact-based environments by intelligent agents, and consequently the reasoning techniques that such agents may adopt to take advantage of these environments. First steps in this direction, with respect to agents with BDI architectures, have been described in [51, 48].

The MERCURIO project aims at putting forward an extension proposal for the FIPA ACL standard, where the FIPA ACL-based communication is integrated with forms of interactions, that are enabled and mediated by the environment. This will lead to an explicit representation of environments as first-class entities (in particular endogenous environments based on artifacts) and of the related model of actions/perceptions. Furthermore we will formulate an improved version of the MAS programming language/framework JaCa, where we plan to integrate the agent-oriented programming language Jason, which is based on a BDI architecture, with the CArtAgO computational framework. This result will extend the work done so far in this direction [55, 57].

In MERCURIO we will implement a prototype of the reference infrastructural model defined by the project. The prototype will be based on the development and integration of existing open-source technologies including JADE [4], the reference FIPA platform, CArtAgO [1], the reference platform and technology for the programming and execution of environments, and agent-oriented programming languages such as Jason [5] and 2APL [27]. The software platform will include implementations of the “context sensitive” ontology alignment algorithms developed in MERCURIO. The algorithms will be evaluated against standard benchmarks and also against the case studies devised in MERCURIO.

Aside from the effects on research contexts, we think that the project may give significant contributions also to industrial applicative contexts, in particular to those companies working on software development in large, distributed systems and in service-oriented architectures. Among the most interesting examples are the integration and the cooperation of e-Government applications (services) spread over the nation. For this reason, MERCURIO will involve some companies in the project, and in particular in the definition of realistic case studies against which the project’s products will be validated. As regards (Web) services, some fundamental aspects promoted by the SOA model, such as autonomy and decoupling, are addressed in a natural way by the agent-oriented paradigm. Development and analysis of service-oriented systems can benefit from the increased level of abstraction offered by agents, by reducing the gap between the modeling, design, development, and implementation phases.
Acknowledgements

We thank S. Mantix for the valuable support and helpful discussions.

References

7. OWL. http://www.w3.org/TR/owl-features/.


49. M. Piunti, A. Ricci, O. Boissier, and J. F. Hübner. Embodied Organisations in MAS Environments. In L. Braubach, W. van der Hoek, F. Petta, and A. Pokahr, editors,


Contextual Integrity and Privacy Enforcing Norms for Virtual Communities*

Yann Krupa, Laurent Vercouter

Laboratory for Information Science and Technology (LIST),
ISCOD team
École des Mines de Saint-Étienne
Saint-Étienne, France
{krupa,vercouter}@emse.fr

Abstract. Contextual Integrity has been proposed to define privacy in an unusual way. Most approaches take into account a sensitivity level or a “privacy circle”: the information is said to be private or public and to be constrained to a given group of agents, e.g. “my friends”. In the opposite, Contextual Integrity states that any information transmission can trigger a privacy violation depending on the context of the transmission. We use this theory to describe a framework that one can use in an open and decentralised virtual community to handle privacy in a socially enforced way. This paper describes a framework, in which we can formally describe privacy constraints, that are used to detect privacy violations according to the Contextual Integrity theory. This framework is part of an ongoing work aiming at applying social control to agents that handle the information, so that malicious agents are excluded from the system.

1 Introduction

Most of the works on privacy focus on security as a means of preserving privacy, either by using a central authority that controls the information access[2,3], cryptography[11], or by using trusted computing techniques[8,6]. Some other views[9,4] aim at designing some preferences that users can attach to the data they “own” without taking into account the possibility of deception by other agents.

While a central authority may be a good solution for a range of applications, it is not possible when working in a decentralized and open system with autonomous agents. Therefore, solutions like Purpose Based Access Control[3] cannot be applied.

One of the problems with Digital Right Management and Trusted Computing measures in general, is that they are very constraining. They impose the use of heavy infrastructure or limit the possibilities of information exchange. These constraints, if they are unacceptable for the users, lead them to interact outside the system that is provided, making every implemented security feature inefficient.

Social regulation is another approach where it is physically possible that violations occur in the system. However, users are observed by the society (usually

*position paper
their neighbours) that can spot them and socially exclude them by ostracism if they commit violations.

So far, very few works consider privacy preserving under the social angle. Yet, it is a prominent problem in applications such as social networks, virtual communities and multi-agent systems, where a social framework will cope naturally with all the social components already working in these systems like trust, reputation, roles for example.

Our work tackles the problem of privacy by using social control in decentralized and open multiagent systems. It is based on Nissenbaum’s “Contextual Integrity” theory[7] which defines privacy in a socially relevant way. Therefore it is possible to assess privacy violations from the agent point of view, and apply a social control relying on available social mechanisms, such as the use of trust management techniques, to prevent further violations. Privacy violations will be reduced without requiring a central authority or invasive security measures.

Contextual Integrity states that any information, as inoffensive as it could seem, can potentially harm a set of agents. It means that Contextual Integrity does not make assessment towards the degree of sensitivity of a given information. All information is regarded as evenly sensitive/insensitive. We call the set of agents that can be harmed by an information, its Targets. We say that an agent is harmed by an information if it makes the agent lose any kind of resource e.g. time, reputation, acquaintances, role. An agent sending information is called Propagator and the agent receiving the information is called Receiver. During the different moments of the process and depending on the information, those attributions may change from one agent to another.

The goal of our work is to provide means to a propagator to use logical reasoning and trust mechanisms to make assessments about a further transmission: “will the transmission of information i to agent z be a violation of contextual integrity?” A receiver should also be able to do the same process when receiving an information: “was the reception of information i from agent a a violation?”. If a violation is detected, social sanctions are thrown against the violating agents.

This paper describes this ongoing work, it proposes a framework in which we can formally describe privacy constraints according to the Contextual Integrity theory and norms in order to enforce these constraints. This framework is then used to detect the occurrence of privacy violations. The sequel of this article is organized as follows. Section 2 presents Nissenbaum’s Contextual Integrity theory and how we interpret it to build appropriateness laws. The characteristics of the application that we consider, virtual communities, is described in section 3 and it is formally described in order to be able to detect automatically privacy violations. Then, a set of privacy enforcing norms are defined in order to give a roadmap for agent’s behavior in section 4. Finally, section 5 shows how these social mechanisms are used to prevent and punish privacy violations on a sample application, and we conclude the paper in section 6.

2 Contextual Integrity

In this section we present the theory of Contextual Integrity[7] and introduce the concept of appropriateness extracted from this theory.
2.1 Original Works

In some approaches[8] privacy is viewed as binary (either the information is private or not). Other models consider different levels of privacy[2] or circles[4], whereas contextual integrity focuses on the appropriateness of a transmission, or use of information. Every information is potentially sensitive.

In order to have a complete description of the foundations of the theory, the reader should refer to the original article[7]. Here we will only focus our work on the concept of “violation”. Nissenbaum says that “whether a particular action is determined a violation of privacy is a function of :

1. the nature of the situation/context
2. nature of the information with regard to the context
3. roles of agents receiving the information
4. relation of agents to information subject
5. terms of dissemination defined by the subject”

Those ideas are way too vague to be used as-is in a software application, we define hereafter a more precise definition of the concepts.

2.2 Appropriateness

We use the term appropriateness to define the set of laws that makes a transmission inappropriate (i.e. will trigger a violation of privacy) if one of these laws is violated. The term “Appropriateness” is inspired by[1].

We use the term “target” instead of Nissenbaum’s term “subject”. A subject is directly related to the information, while a target may not even appear in the information. For example, if the information is a picture of Mr X’s dog doing bad things on X’s neighbour’s grass, the subject is the dog but the target, the one that can be harmed by the disclosure of the picture, is X. Therefore, we think that considering the target instead of the subject is more versatile.

We can then define a flow as appropriate if all of the following conditions hold, and inappropriate if one of the conditions does not hold (numbers in parenthesis refers to the corresponding statement in Nissenbaum’s definition above):

1. Transmission context must correspond to the information nature (1+2),
2. Agent must have a role within the transmission context (3),
3. Agents must not have incompatible relations with target\(^1\) (4),
4. The target’s preferences must be respected (5)

If a flow is inappropriate then there is a privacy violation. Here we can see the point of this approach: an information is not “public” or “private”, every information can trigger a privacy violation if it is used inappropriately.

Thereafter, we illustrate the 4 statements of appropriateness with examples:

\(^1\) This item is a work in progress and is not taken into account in the following parts.
1. In the large sense, the context of a transmission can be seen as the environment where and when the transmission takes place. In our framework, for simplification means, we will say that the context of a transmission is declared by the propagator. A context corresponds to the information if it reflects the nature of the information, e.g.: personal health information corresponds to medical context.

2. Agents participating in the transaction should have a role associated to this context[1]. For example, a medical doctor has a role belonging to the medical context.

3. Sometimes, it is not sufficient that the agent has some roles belonging to the context of the transmission. Because some relations between the target of the information and the agent receiving the information may be inappropriate. For example, consider the case of an agent A who has an illness, and an agent B who is both a medical doctor and A’s boss. It may be inappropriate for B to know A’s disease because those agents are having an “out of context” relationship (hierarchical relationship).

4. If one of the targets of the information specifies preferences regarding the propagation of the information, it is inappropriate to violate those preferences.

As appropriateness has been defined it is now necessary to define the kind of application we consider, information transmission in virtual communities. Afterwards, we propose a formalism of appropriateness to be used in this kind of application.

3 Framework

This section presents the application domain and all the components needed to handle contextual integrity as defined in the previous section, as well as a formalism of appropriateness.

3.1 Privacy Preservation in Virtual Communities

In several types of virtual communities, such as social networks or virtual enterprises\(^2\), users communicate and share information using software systems that support the community. These applications raise a difficult problem of privacy preservation. On the one hand, it is the main goal of these communities to enable communication so that users can easily send information to their contacts. On the other hand, as it is stated by the contextual integrity theory, each piece of communicated information may result in a privacy violation. Indeed, if we consider the case of a virtual enterprise, the community includes users with different hierarchical roles, belonging to different services but also different enterprises. It is obvious that all information should not be sent to other users without analysing the nature of information and of the concerned users. The same case

\(^2\) A virtual enterprise is a temporary collaborative network of enterprises made in order to share resources and competences.
occurs in professional or personal social networks in which users’ contacts can be her colleagues, siblings, friends.

The goal of our work is to specify a software assistant agent that is able to help a user to preserve privacy in a virtual community. The assistance is both to preserve the user’s privacy by providing advices when an information is communicated (should he send this information or not to a given contact?), and to preserve the other users’ privacy by detecting when a privacy violation occurred and should be punished. This paper describes the first steps of this ongoing work by defining a language to express privacy constraints and means to detect privacy violations.

The virtual community that we consider has the following characteristics. It works as a peer-to-peer network, meaning that information is exchanged by a communication between one sender and one receiver. Moreover, it is a decentralized and open system. It is thus impossible to define a centralized control that relies on a global and complete perception of communications. We have chosen a system with these features to be as general as possible. By proposing a local assistance to users, the assistant agent can be used both in centralized and decentralized systems and it does not constrain the system scalability. The choice of peer-to-peer communication is also general enough to be able to represent other kinds of communications. For instance, if we want to consider a social network in which information is exchanged by publishing it on a page or a “wall” readable by the user’s contacts, it can be represented by several one-to-one communications.

In order to be able to define privacy preservation according to contextual integrity, we need to introduce two concepts in the virtual community: context and role. The context describes the situation in which an information is exchanged. Examples of context are “Dave’s work”, “John’s family”, “health”. Roles are defined within a context and attached to users. Examples of roles in the three contexts mentioned above are respectively “Dave’s boss”, “John’s father”, “medical doctor”. There can multiple roles per context. In this paper, we assume that users’ roles and their corresponding contexts are provided by organisational entities that act as repositories. These entities are able to return the role associated to a specific user and the context associated with a specific role. For this purpose, it is possible to use organisational multiagent infrastructures[5].

These concepts are useful to be able to express rather fine rules for Contextual Integrity. We use them in the next subsections to allow the assistant agent to reason on privacy violations.

### 3.2 Message Structure

Users exchange information encapsulated in a message. Information is raw data. We don’t make assessment about the structure of the information and leave it free. A message encapsulates information plus meta-information described below.

First, from a given information, can be computed a unique reference that allows to refer unambiguously to the information without carrying itself the information (Hash algorithms like Message Digest[10] can be used).
Then, the message adds the following meta-information:

- Context Tags: tags referring to the context of the information
- Target Tags: tags referring to the targets of the information
- Privacy Policies: policies expressing preferences regarding further distribution of information
- Transmission Chain: a chain of transmissions that allows to keep track of the message path in the system

Each of these components may be digitally signed by agents that wish to support the meta-information accountability. When signing a meta-information an agent engages his responsibility. The semantics that relies behind the signature is a certification: i.e. the agent that signs the context tag “medical context” certifies that the information is about medical context. Therefore, it is very important that a meta-information, even if it can be detached from the information (which is possible), cannot be reattached to another information. We prevent that from happening by including the information hash before signing. Signatures are formed by a name and a signature (RSA signature for example[11]). The transmission chain allows to keep track of the message path among the agents. Every agent is required to sign the chain before propagating a message, an agent adds his signature including his own name and the name of the receiver of the message.

3.3 Primitives

To allow the agent to recover data regarding the concepts described earlier, like the meta-information or the roles of agents, we need to provide the agents a set of logical primitives. These primitives can then be used to express constraints about transmission of information.

1. Primitives based on meta-information:
   - \texttt{information}(M,I). Means that I is the information\(^3\) encapsulated in message M.
   - \texttt{contexttag}(C,A,M). Means that C is the context tag for message M signed by agent A.
   - \texttt{targettag}(T,A,M). T is the target tag for message M, signed by A.
   - \texttt{policy}(P,A,I). There is a policy P signed by agent A for information I.

2. Primitives based on transmission roles:
   - \texttt{receiver}(X,M). Agent X is receiving the message M.
   - \texttt{propagator}(X,M). Agent X is sending the message M.

3. Primitives based on agent beliefs:
   - \texttt{target}(X,I). The agent believes that agent X is targeted by the information I.

\(^3\) The primitives are referring to an information I or a message M. This is because some primitives will be specific to a given message M, and some others will be common to all messages containing the same piece of information I.
The agent believes that the preferences expressed by policy $P$ are respected for the information $I$.

- $\text{context}(C,I)$. Means that the agent believes that $C$ is the context of information $I$.
- $\text{role}(A,R)$. The agent believes that Agent $A$ has the role $R$.
- $\text{rolecontext}(R,C)$. The agent believes that role $R$ belongs to context $C$ (role “surgeon” belongs to Medical context).
- $\text{link}(X,Y)$. The agent believes that agent $X$ is capable of communicating with $Y$.

Now, based on this primitives, we are able to express preferences or norms.

### 3.4 Appropriateness Laws

Our goal is to obtain some simple laws that agents can rely on to be able to decide if a given transmission of information should be seen as a violation or not.

These appropriateness laws are thereafter abbreviated as A-laws.

This is the definition of the A-laws we propose in Prolog-like code:

- Context declared by the propagator must be equal to the information context (Fig. 1).
- Receiver must have a role within the transmission context (Fig. 2).

```prolog
fitcontext(C,M):-
  information(M,I),
  propagator(P,M),
  context(C,I),
  contexttag(C,P,M).

fitrole(C,M):-
  receiver(Rc,M),
  role(Rc,R),
  rolecontext(R,C).
```

**Fig. 1. fitcontext**  
**Fig. 2. fitrole**

- The target’s preferences must be respected:
  - In the case there is no policy defined by a target then $\text{fitpolicy}(M)$ holds (Fig. 3).
  - If there is a policy defined by the target, the agent must respect it (Fig. 4).

Therefore, a transmission is defined as appropriate for a message $M$ if the following formula holds:

```prolog
appropriate(M):-
  fitcontext(C,M),
  fitrole(C,M),
  fitpolicy(M).
```

If the definition above does not hold, then we can say that the transmission is inappropriate, there is a violation of the contextual integrity.

\[\star\] is the negation-as-failure in Prolog.
3.5 Policies

The A-laws define what is appropriate or not in a general point of view, but targets can define policies (preferences) in order to constrain the information. These preferences are defined for a given information by a given agent who signs the policy. In the system, it is not possible to insure that a policy cannot be detached from the information it is referring to, i.e. an agent may erase the policy at some point. But it is ot possible to reattach a policy to another information, because the policy is signed, and contains a pointer to the information it refers to.

A policy is composed by several statements. A statement is composed by several primitives from the ones described in section 3.3 and by a type of statement that can be:

- forbidden(I):-
  Declares a situation that should not occur within a transmission of information I.
- mandatory(I):-
  Declares a situation that has to occur within a transmission of information I.

A given policy is fulfilled if none of its forbidden statements holds (if one holds, then it is unfulfilled) and one of its mandatory statements holds[1].

An example of policy for a given information identified by ‘info99’ is given below. It is composed by two forbidden statements (do not send data to an agent who has a common contact with the target AND don’t send data to the target) and one empty mandatory statement.

\begin{verbatim}
forbidden(info99):-
  information(M,info99),
  receiver(X,M),
  target(T,info99),
  link(X,Z),
  link(Z,T).
\end{verbatim}

\begin{verbatim}
forbidden(info99):-
  information(M,info99),
  receiver(X,M),
  target(T,info99),
  link(X,Z),
  link(Z,T).
\end{verbatim}

A statement is composed by a conjunction of primitives, therefore the disjunction is expressed by defining multiple statements of the same kind. This is why only one mandatory statement is required to validate the policy and one forbidden to invalidate it.
In order to test the primitive $\text{policyvalid}(P, I)$, an agent adds to his memory all the statements contained in policy $P$ (we suppose here that we have a primitive $\text{addpolicy}(P)$ to do just that):

\[
\text{policyvalid}(P, I):= \\
\text{addpolicy}(P), \\
\neg \text{forbidden}(I), \\
\text{mandatory}(I).
\]

## 4 Privacy Enforcing Norms

As shown in the previous sections, we need the agents to check the transmissions, to be able to see if there are violations and punish the responsible. This section propose a set of norms that defines what should be the behavior of a compliant agent in the system. Then it describes the punishment mechanisms and finally discusses the inherent problems regarding the subjectivity of beliefs.

### 4.1 Definition

The basic component of the system is the set of A-laws, that express Contextual Integrity violation. But the keystone of the system are the Privacy Enforcing Norms (PENs), defined in this section, that instruct the agents to respect the A-laws and punish those who do not.

The PENs are the following:

1. Respect the Appropriateness laws
2. Sign the transmission chain before sending
3. Do not send information to untrusted agents
4. Delete information from violating or untrusted agents
5. Punish agents violating these norms (this one included)

The first norm (PEN 1) that we propose is meant to protect the A-laws from being violated: “Respect the Appropriateness laws”.

From our point of view, every agent must take responsibility when doing a transmission. Thus we define a norm stating that every agent has to sign the transmission chain (in order to backtrack the potential violation to its source). We also consider that sending information to an agent while knowing that he will commit a violation, is a violation itself. Two new norms are then defined: “Sign the transmission chain before sending (PEN 2); Do not send information to untrusted agents (PEN 3).” The PEN 3 also implements the social punishment, because agents will stop communicating with these untrusted agents.

The fourth norm aims at minimizing the violations by deleting information received from unreliable agents (PEN 4).

Norms that the agents should respect have been defined, but we want to be sure that the agents in the system will punish those who do not respect the norms, henceforth punishing those that do not punish agents not respecting the norms. This last norm (PEN 5) insures consistency of the PENs, because an
agent that decides to violate a norm will be punished, others will stop trusting
him and eventually he will become socially isolated.

Therefore norms are not enforced by the system but by the agents themselves
and agents refusing to enforce the norms will be punished by other agents. For
now, the punishment is implemented as a social punishment: an agent witnessing
a violation has to send a message to all of its contacts stating the details of
this violation. The following section gives more details about this punishment
mechanism.

4.2 Punishment

When an agent detects a violation of the PENs, PEN 5 states that he has to
send a punishment message. This message is meant to describe the violation so
that other agents can punish the culprit. The message has the same structure
than all the messages in the system: information and meta-information. Here
the information part contains:

– The meta-information of the original message source of the violation
– A description of the violation using the primitives of section 3.3

Sending the meta-information of the original message is useful to provide
evidence to other agents that may not believe that there was a violation. The
advantage of sending only the meta-information is that the agent will not trans-
mit the information itself (which could in turn trigger a violation and so on).

The violation is described using the primitives, and the PEN that has been
violated. For instance, if the PEN 3 has been violated by Bob, the following
primitives will be sent:

\[
\text{pen3violation}(\text{Bob, mess45}),
\text{receiver}(\text{John, mess45}),
\text{propagator}(\text{Bob, mess45}),
\text{untrustworthy}(\text{John}).
\]

These primitives will be handled and verified by the receiving agent. If the
agent agrees with every primitive in the argument, then he can propagate the
punishment message and punish the culprit by revising its trust. There are sit-
uations where the agents may not have the same beliefs, e.g. John may or may
not be trustworthy depending on the agent making the assessment.

4.3 Discussions on Subjectivity

Some of the PENs are very subjective, because they are based on beliefs. There-
fore 2 given agents in the system may not have the same belief and interpret the
norms differently. For instance, two agents, $A$ and $B$ may have different beliefs
regarding agent $X$ trustworthiness: $A$ trusts $X$ but $B$ does not. Now $A$ sends a
message to $X$ who in turn sends the message to $B$. In the transmission chain,
$B$ is able to see that the transmission occurred between $A$ and $X$, which violates
norm 3. Going back from $A$ point of view, it would not be fair to be punished
for this transmission as $X$ seems trustworthy for him.
B witnessed a violation so he has to send a punishment message. The punishment message has to argue about the punishment. More than just saying “A does not respect the norms”, B makes a message stating that “A violated the third norm because B believes that X is untrustworthy and A sent a message to X”. The agent receiving this violation message is going to check these statements and if he agrees, he can revise his trust level towards A.

Along with the violation description, the punishment message contains the meta-information of the original message. This allows other agents to check the PENs and violation description. For instance, it will allow agents to check that A did sent the information to X by looking at the transmission chain contained in the meta-information.

4.4 Usage

This section describes how the agents are meant to protect privacy using the tools provided in the previous sections. As it is said in the introduction, our goal here is to handle privacy from the agent perspective to minimise the number of violations in the whole system.

There will be two main situations:

- Receiving: When the agent receives an information: “Does the agent that sent me this information made a violation by sending it to me?”
- Propagating: When the agent is about to send information: “Am I going to make a violation if I send the information to this agent?”

Trust In the framework presented in this article, agents may perceive things differently. If we take a closer look at the primitive context(C,I) described earlier, for instance, it is stated that it means that the “agent believes that C is the context of information I”. Therefore, some agent X may believe for a given information that the context is O, and another agent Y may believe that the context is P. This situation can happen because the agents are autonomous and have beliefs that can be different from one to another. As they have different beliefs, some agent may think that a given transmission is inappropriate, and another may think that it is not. Because of this uncertainty, when an agent detects a violation, he is not able to be sure that the other agent made the violation on purpose, therefore it will be unfair to kick him directly from the system. This is where trust comes in, this kind of “soft security” is able to cope with detection errors while still being able to exclude the ones that make violations. Trust is one of the main components to decide who is reliable or not for handling our information. If someone is untrustworthy, we are not willing to send him any piece of information.

The trust management component of agents is not yet implemented and is being defined in our current ongoing work. We will probably use an adaptation of existing computational trust models for multi-agent systems such as LIAR[13] or Repage[12].
Receiving When the agent is receiving a message, he has to check if the transmission that just occurred is a PEN violation or not. First, the agent has to check the A-laws to see if the transmission is appropriate (PEN 1), as described in section 2.2. To do that, the agent will have to infer multiple things, for example: who is the target of the message? what is the context of the message? This is possible either by using personal knowledge, by using the context tags and target tags or by analysing the information directly. As the context tags (and target tags) are signed, it is possible to trust the agent that signed the given tag, to come to believe that this context tag corresponds to the context of the information.

If the agent detects a PEN violation, he sends a “punishment message” to other agents.

Finally, the agent readjusts the trust level he has towards the propagator that just made the violation.

Propagating This second situation happens when the agent is about to send information. Before sending, it is necessary to attach to the information all possible meta-information:

- If the agent can identify the target of the information (by using knowledge or information analysis), he adds a target tag for target Z that he signs. This states that the agents confirms that the target of the information is Z.
- If the agent is able to determine the context of the information (by using knowledge or information analysis), he adds and signs a context tag.
- If the agent is the target, he can specify some restrictions regarding further dissemination of the information, in this case, he adds a policy that he signs.
- The agent also signs the transmission chain to insure PEN 2.

Then, the agent should make all PEN assessments towards the receiver:

- Does the agent violates the A-laws (PEN 1) by sending the information to the receiver? An agent never violates A-laws, except if he is malevolent or ignorant, which in both cases, will be punished by other agents.
- Does the agent trust the receiver? (PEN 3) If he is untrustworthy, it means that he has probably made some privacy violations in the past. As the agent aims at protecting the information he holds, he only sends to the ones he trusts.
- And so on with the other PENs.

At the end, the agents send information from one to another, checking before sending and after receiving if some violation has occurred. When violations are detected, agents send “punishment messages” to their contacts, so that others become aware of the violation that occurred. Eventually, agents that make violations will be socially excluded from the system, because no agents communicate with untrustworthy agents.

5 Sample Application

Our aim here is to define a sample application to show how all the framework components instantiate on this application.
5.1 Photo Sharing

The application that we consider here is a photo sharing social network. Basically, users can share pictures with their contacts who can, in turn, share again those pictures with their own contacts and so on. We provide the users with an assistant agent that will do all the assessment described before to inform the user of any violation. The final decisions lies in the hands of the user, the assistant does not take any decision.

In this system, the pictures are the information that is exchanged.

5.2 Primitives Instantiation

Some of the primitives we defined earlier need to be specified for this application. The primitives based on meta-information always remain the same, because the nature of the meta-information does not change. So do the primitives for transmission roles.

We can explain in more detail the primitives based on agent beliefs because the way they are inferred is what is interesting here:

- $\text{context}(C,I)$ For the agent to believe that $C$ is the context of information $I$, there are alternative solutions:
  - Look if there is a context tag emitted by a trusted agent
  - Analyse the picture to find its context (using image analysis techniques)
  - Ask the user attached to the agent to determine the context of the picture

- $\text{target}(X,I)$ The same process can be used for the agent to believe that $X$ is the target of $I$:
  - Look if there is a target tag emitted by a trusted agent
  - Analyse the picture to find if the target is on the picture
  - Ask the user attached to the agent to determine the target of the picture

- $\text{link}(X,Y)$ By analysing the transmission chain in the meta-information, the agent can discover links between other agents.

- $\text{knew}(X,I)$ Using the same technique, the agent can extract from the transmission chain the list of agents that received the information in the past.

- $\text{role}(A,R)$ The agent asks the organisational infrastructure to know the possible roles of $A$.

- $\text{rolecontext}(R,C)$ The agent asks the organisational infrastructure to know the possible roles fitting in context $C$.

- $\text{policyvalid}(P,I)$ The agent infers on his belief base to see if the policy is valid as explained in section 3.5.

With the primitives instantiated, it is easy to check the policies, the A-laws and all needed components. In the next section we show an example of what happens in the application.
5.3 Use Case

Alice wants to share a picture with Bob. The target of the information is James, who is in an awkward position on the picture. Some of James’ friends already had this information before, therefore, there are tags describing the context as “James friends” and the target as “James”. No policy has been attached. The unique identifier of the information is “pic254”. The message is identified by “mess412”.

When Alice clicks on the button to send the picture to Bob, the assistant agent checks the PENs:

- PEN 1: Does the agent violates the A-laws by sending the information to the receiver? This is the instantiation of the laws described in section 3.4:
  - The declared context is set by the agent, so the declared context fits the context the agent believes to be the real one, the following formula holds:
    \[
    \text{fitcontext}('James friends',\text{mess412}):= \\
    \quad \text{information}(\text{mess412}, \text{pic254}), \\
    \quad \text{propagator}('Alice',\text{mess412}), \\
    \quad \text{context}('James friends',\text{pic254}), \\
    \quad \text{contexttag}('James friends','Alice',\text{mess412}).
    \]
  - The assistant agent is not able to find a role for Bob that fits into the context “James friends”, the formula does not hold:
    \[
    \text{fitrole}('James friends',\text{mess412}):= \\
    \quad \text{receiver}('Bob',\text{mess412}), \\
    \quad \text{role('Bob',?)}, \\
    \quad \text{rolecontext(?,'James friends')}. \\
    \]
  - No policies were defined, therefore, the first \text{fitpolicy}(M) statement holds (no policy exists for none of the target of the information).

The following Appropriateness formula does not hold, because Bob is not a friend of James (the target):
\[
\text{appropriate(\text{mess412})}:= \\
\quad \text{fitcontext}('James friends',\text{mess412}), \\
\quad \text{fitrole}('James friends',\text{mess412}), \\
\quad \text{fitpolicy}(\text{mess412}).
\]

Beyond this point, the assistant agent knows that the transmission will be inappropriate, and therefore violates the PENs. Anyway, he asks the user (Alice), what to do: continue or abort the transmission?

Alice wants to continue. The message containing the picture and meta-information is sent to Bob.

Bob’s agent handles the information by checking the PENs:

- Does the message violates contextual integrity? Bob’s agent runs here the same test that Alice’s agent did (using his own beliefs). As Bob is not a friend of James, no roles fits in the context “James friends” and a violation is therefore detected.
Bob's agent adjusts his beliefs, he does not trust Alice anymore because it is not the first time that Alice deceives Bob. He sends to all his contacts a “punishment message” containing the meta-information (context tags, target tags, transmission chain) and the following description:

\[
\begin{align*}
\text{penviolation}(Alice, mess412), \\
\text{information}(mess412, pic254), \\
\text{context}(C, mess412), \\
\text{\textbackslash+ fitrole}(C, mess412).
\end{align*}
\]

Dave’s agent is one among those who receives this message. Dave was about to send a message to Alice, when he clicks the “send” button, his agent checks the PENs. Then PEN 3 forbids to send a message to an untrusted partner. Dave’s agent warns him that Alice is untrustworthy and that the transmission will violate the PENs. Users stop communicating with Alice because of the violation she made. Alice is now socially excluded, she is yet still in the system but nobody keeps communicating with her.

The example is a little bit hard on Alice in order to show the power of social exclusion. Normally, it will take multiple violations for someone to be excluded from the system and forgiveness could occur after a certain time.

6 Conclusions

The framework we presented in this article allows to protect users privacy in a decentralised and open system when it is not possible to apply computer security approaches. Based on Nissenbaum’s Contextual Integrity theory, we propose an approach using appropriateness laws that defines what is an appropriate information transmission (and therefore, what is inappropriate). Primitives are defined to express these laws and to allow agents to define preferences over the transmission of some specific information.

Agents in the system play both roles of actors and judge: they transmit information, and they detect violations. Agents also inform others when they spot a violation, so that the violating agents are excluded of the system. This behavior is directed by the Privacy Enforcing Norms (PEN) described in this article.

There are still some unsolved problems in the system, that may prevent it from working correctly:

- The trust related problems: “what happens if there are too many malevolent agents in the system?”
- “Journalist Problem”: “what happens if an agent decides to sacrifice himself to become a relay for information that violates privacy?” (the original source is never punished, only the journalist).
- Reputation Paradox: Information about reputation is libellous in a way, so it can generate privacy violation. But at the same time, it is required for maintaining information regarding agents that make violations.
In our future works, we will integrate the trust mechanisms directly in the decision process, i.e. decompose the primitives that rely on trust in predicates. We will, at the same time, investigate the problems related to the subjectivity and related to strategic manipulation (agents sending fake violation messages for example). Then an application will be developed to probe the system in the real world by providing assistant agents to users.

References