Formalizing Organizational Constraints

A Semantic Approach

M. Birna van Riemsdijk[1] Koen V. Hindriks[1] Catholijn M. Jonker[1] Maarten Sierhuis[2]
Delft University of Technology, Delft, The Netherlands[1]
Carnegie Mellon University, California, USA[2]
{m.b.vanriemsdijk,k.v.hindriks,c.m.jonker}@tudelft.nl[1]
maarten.sierhuis@sv.cmu.edu[2]

ABSTRACT

An *organizational modeling language* can be used to specify an agent organization in terms of its roles, organizational structure, norms, etc. Such an organizational specification imposes constraints on agents that play roles in it, and the agents are expected to take this into account when deciding what to do. This means that agents need to have a basic understanding of what it means to comply with organizational constraints. For this, it is essential that these constraints are precisely specified. In this paper, we address this in the context of the MOISE$^+$ organizational modeling language. We define a semantic framework for MOISE$^+$ MAS and an accompanying linear temporal logic (LTL) to express its properties. We analyze which constraints MOISE$^+$ imposes on agents, and investigate how these can be made precise in LTL. We show that multiple interpretations of constraints are sometimes possible, and explore the space of possibilities. These analyses demonstrate the need for a rigorous specification of organizational constraints, and provide the foundations for the development of agents that understand how to function in a MOISE$^+$ MAS.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; F.3.2 [**Logics and Meaning of Programs**]: Semantics of Programming Languages

General Terms

Theory, Languages

Keywords

Organizational Modelling Languages, Organization-Aware Agents, Temporal Logic

1. INTRODUCTION

An important line of research in the multi-agent systems (MAS) field that has received increasing attention in the last years, is to assign an organization to the MAS with the aim of organizing and regulating it. Assigning an organization to a MAS can be done by developing an *organizational specification* in an organizational modeling or programming language (see, e.g., [5, 1, 14, 4, 6]). An organizational specification abstracts from the individual agents that will eventually play the roles in the organization. It may define the structure of the agent organization in terms of roles and the relations between roles, and specify the norms (e.g., obligations and prohibitions) that are to be followed by the agents of the MAS. Organizing a MAS should make the agents more effective in attaining their purpose, or prevent certain undesired behavior from occurring. An organizational specification achieves this by imposing *organizational constraints* on the behavior of agents that function in the organization.

Agents that operate in such an organized MAS are expected to take these organizational constraints into account when deciding what to do. For example, if an agent plays a role, this typically comes with obligations that are to be adhered to. Agents should be aware of this and take this into account when deciding on action, if they are to operate effectively and flexibly in the organization. Agents that are capable of such organizational reasoning and decision making are called *organization-aware agents* [20, 18]. Organization-aware agents should be contrasted with agents that have been designed to function in a particular organization and that do not reason about the organizational specification. For such agents, it will be more difficult to adapt their behavior to changes in the organizational specification. That is, an important advantage of organization-aware agents is added flexibility due to the fact that they are able to understand the organizational specification.

Our research objective is the development of languages and techniques for organization-aware agents. An essential step towards this is specifying clearly what an organization expects from agents, i.e., what the organizational constraints are. Ambiguity or unclear specifications of such constraints may at best result in innocent misbehavior on the part of the agents but at its worst may result in a dysfunctioning organization. Moreover, without a precise specification of organizational constraints it is not clear what to aim for when developing (languages and techniques for) organization-aware agents.

In this paper, we investigate organizational constraints in the context of the well-known MOISE$^+$ organizational modeling language [13, 11, 14]. We introduce MOISE$^+$ in Section 2. MOISE$^+$ does not come with a comprehensive formalization of all organizational constraints. Nevertheless, some aspects are formalized in [11], and [12] formalize some constraints by expressing them in a normative programming language with formal semantics. Our approach to making organizational constraints precise is to define a formal semantic framework for MOISE$^+$ MAS and an accompanying linear temporal logic (LTL) to express its properties (Section 3). We discuss which constraints MOISE$^+$ imposes on agents, and analyze them by making them precise in LTL (Sections 4 and 5). We show that multiple interpretations of constraints are sometimes possible, and explore the space of possibilities by formalizing

Cite as: Formalizing Organizational Constraints: A Semantic Approach, van Riemsdijk, Hindriks, Jonker, and Sierhuis, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

different variants in LTL and investigating their properties. These analyses demonstrate the need for a rigorous specification of organizational constraints, and provide the foundations for the development of organization-aware agents that function in a MOISE$^+$ MAS. We conclude the paper and discuss future work in Section 6.

2. MOISE$^+$

In this section, we present a general overview (Section 2.1) and more detailed specification (Section 2.2) of MOISE$^+$.

2.1 Overview

The MOISE$^+$ organizational modeling language [13, 11, 14] specifies an organization in terms of a *structural* dimension using the notions of roles and groups, a *functional* dimension that describes how global collective goals should be achieved by means of so-called schemes, and a *deontic* dimension expressing permissions and obligations for roles, related to the achievement of (sub)goals. Together, these dimensions form an *organizational specification*, which imposes *organizational constraints* on a MAS, for example by specifying obligations to achieve particular goals when an agent plays a certain role.

Given an organizational specification, it is up to the agents to operationalize it by entering the organization and playing roles in it. In MOISE$^+$, the run-time information regarding, e.g., which agents take part in the organization and which agent has adopted which role, together with the organizational specification is called an *organizational entity* (*OE*). Agents can change the organizational entity by executing *organizational actions* such as adopting a role. MOISE$^+$ comes with organizational middleware S-MOISE$^+$ [14]. The middleware maintains the organizational entity and can execute organizational actions if requested so by agents. The middleware can also prevent the execution of organizational actions if their execution would violate organizational constraints.

Figure 1 depicts the most important organizational actions, and loosely illustrates when they can be executed. The figure also depicts hard and soft constraints, which we will discuss in the sequel. It will become clear that hard constraints apply as soon as an agent enters the organizational entity, and soft constraints are relevant once an agent has adopted roles and once scheme instances have been created. The rest of the figure should be read as follows.

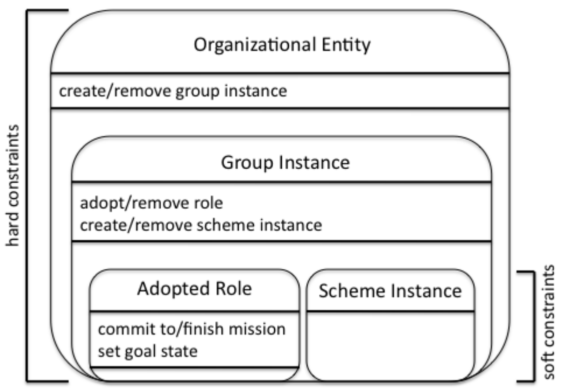

Figure 1: Participating in a MOISE$^+$ Organization

Once an agent has entered the organizational entity, it can create[1] *group instances*. These instantiate the groups that are specified in the structural specification. A group is an element of a structural specification that is connected to roles and to other groups through a subgroup relation. A group instance is part of an organizational entity. Once a group instance has been created, agents can *adopt roles* in the group instance, and create *scheme instances* for which the group instance is responsible.[2] A group instance is well-formed once enough agents are playing the roles of the corresponding group. A role can only be adopted and a scheme can only be created for a group instance. Group instances thus have to be created before agents can adopt roles, since roles should be adopted in group instances. This also explains why there is no restriction on which agent can create which group instance: it should be possible to create group instances before an agent has adopted a role. The creation of a group instance can be viewed as an invitation to other agents to form a group.

A scheme specifies how global goals can be achieved by decomposing these into subgoals, and specifies how these (sub)goals are distributed over agents using the notion of *mission*. A mission consists of one or more (sub)goals, and the deontic specification relates roles to missions by defining to which missions an agent playing a certain role is obliged or permitted to commit. Once an agent has adopted a role in a group instance, it can commit to obliged or permitted missions of scheme instances for which the group instance is responsible. There is thus a difference between the agent being obliged to commit to a mission, and the agent actually making the commitment. The idea is that through committing to a mission, the agent indicates that he "accepts" the obligation and will try to achieve the goals of the mission. Once an agent has reached a goal of a mission in a scheme instance, the agent can notify the organization of this by setting the goal state to "achieved" (or alternatively setting the goal state to "impossible" if the agent believes the goal cannot be achieved).

2.2 Detailed Specification

We now specify the dimensions of the organizational specification in more detail. We make several simplifications for reasons of presentation and refer to [13, 11, 14] for more details. A *structural specification* consists of a set of roles $\mathcal{R}$ with typical element ρ, a set of *acq*uaintance, *com*munication and *aut*hority links $\mathcal{L}$ between roles of the form (ρ, ρ', l) where $\rho, \rho' \in \mathcal{R}$ and $l \in \{acq, com, aut\}$ is the type of the link, a set of compatibility constraints $\mathcal{C} \subseteq (\mathcal{R} \times \mathcal{R})$ that is reflexive and transitive, and that expresses which roles an agent is allowed to play simultaneously, a set of groups $\mathcal{GR}$, and a set of group-role and group-group relations $((\mathcal{GR} \cup \mathcal{R}) \times \mathcal{GR})$ expressing which roles are part of which groups, and which (sub)groups are part of other groups. For all links in $\mathcal{L}$ it holds that an authority link between two roles implies a communication link, and a communication link implies an acquaintance link. For simplicity, we have omitted inheritance between roles.

A *functional specification* describes how global goals should be achieved. It consists of a set of *schemes* $\mathcal{S}$. A scheme consists of a set of goals $\mathcal{G}$, a set of mission labels $\mathcal{M}$, a goal tree $\mathcal{T}$ (containing the goals from $\mathcal{G}$) that describes how to decompose goals into subgoals (using the operators sequence, choice and parallelism, see [13] for details), and a function $m2g : \mathcal{M} \to \mathcal{P}(\mathcal{G})$ that assigns a set of goals to mission labels.[3] The latter function defines the mis-

[1]For each "create" action there is also a dual "remove" action.

[2]Note that the notion of "group" in MOISE$^+$ thus does not directly refer to a collection of agents. It is an element of the structural specification, and only once it is instantiated can agents adopt roles in the resulting group instance.

[3]In [13], missions also have a minimum and maximum number of agents that should commit to them. For simplicity, here we assume the minimum is 1 and the maximum is ∞.

sions of a scheme, which are sets of coherent goals that an agent can commit to. We use *root*(s) to denote the root goal of scheme s.

A *deontic specification DS* describes which missions an agent playing a certain role is obliged or permitted to commit to. The deontic specification consists of a set of permissions and obligations of the form *permission*(ρ, m) and *obligation*(ρ, m), respectively, where $\rho \in \mathcal{R}$ is a role and $m \in \mathcal{M}$ is a mission label.

The set of agent names of agents participating in an organizational entity *OE* is denoted as $\mathcal{N}$, the set of group instances that have been created are denoted as $\mathcal{GI}$, which agent has adopted which role in a certain group instance is represented by a mapping *a2r* : $\mathcal{N} \to \mathcal{P}(\mathcal{R} \times \mathcal{GI})$, the set of scheme instances that have been created are denoted as $\mathcal{SI}$, the scheme instance information is represented by a mapping *si2i* : $\mathcal{SI} \to (\mathcal{S} \times \mathcal{P}(\mathcal{GI}) \times \mathcal{N})$ that maps each scheme instance to its corresponding scheme, responsible group instances, and the agent that created the scheme instance, we represent which agent is committed to which missions in which scheme instance by the mapping *a2c* : $\mathcal{N} \to \mathcal{P}(\mathcal{M} \times \mathcal{SI})$, and the state of a goal in a scheme instance is represented by the mapping *g2state* : $(\mathcal{SI} \times \mathcal{G}) \to \{\textit{unsatisfied}, \textit{satisfied}, \textit{impossible}\}$.

In the sequel, we use the notation introduced here to refer to parts of an organizational entity *OE*, without explicitly stating this each time, e.g., $\mathcal{GI}$ refers to the group instances of *OE*, $\mathcal{SI}$ refers to the scheme instances, and so forth.

3. BASIC FORMALISM

In this section, we introduce the basic semantic framework defining the execution of a MOISE$^+$ MAS (Section 3.1), together with a linear temporal logic (LTL) to express properties of its execution (Section 3.2).

3.1 Semantic Framework

First, we define which components constitute a state of a MOISE$^+$ MAS. We define a state of a MOISE$^+$ MAS as a tuple $\langle OE, \xi, a_0, \ldots, a_n, \mathsf{act}_i\rangle$ where *OE* is an organizational entity, ξ represents the shared environment in which the agents operate, $a_0, \ldots, a_n$ are the states of the agents participating in the MAS where $0, \ldots, n$ are the respective agent names, and act_i is the action act that has been executed by agent i to reach this state from the previous state. The set of agent names $\mathcal{N} = \{0, \ldots, n\}$ of agents participating in the MAS is part of *OE*. We write an underscore instead of act_i if the executed action is not relevant. In the initial state of a MAS we write $\mathsf{act}_i = \epsilon$ to denote that no action has been executed yet. In this paper, we abstract from the internal structure of agents and how these are updated since we specify constraints from the perspective of the organization.

We assume the repertoire of actions Act_i of a participating agent i consists of internal actions (Act_i^{int}), environment actions (Act_i^{env}), organizational actions (Act_i^{org}), and communication actions (Act_i^{com}).[4] The execution of internal actions affects the internal mental state of the agent executing the action, environment actions affect the environment ξ, organizational actions affect the organizational entity *OE*, and communication actions affect the mental states of sender and receiver of the message.

The possible organizational (Act^{org}) and environment (Act^{env}) actions are defined by *OE* and ξ, respectively. We assume these to be mutually disjoint and disjoint from communication actions. We assume for any agent i in the MAS that $\mathsf{Act}_i^{org} \subseteq \mathsf{Act}^{org}$ and $\mathsf{Act}_i^{env} \subseteq \mathsf{Act}^{env}$. We do not further specify the environment and the environment actions, since it is not the focus of this paper (see, e.g., [4] for an example on how this can be done). The organizational actions Act^{org} are those offered by MOISE$^+$ [11], of which we have introduced several in Section 2.

In this paper, the following actions are particularly relevant: adopting and removing a role in a group instance ($\mathbf{adoptRole}_i(\rho, gi)$ and $\mathbf{removeRole}_i(\rho, gi)$), creating a scheme instance from a scheme with a set of responsible group instances ($\mathbf{createScheme}_i(st, gis)$) and removing a scheme instance ($\mathbf{removeScheme}_i(si)$), committing and finishing a mission in a scheme instance ($\mathbf{commitMission}_i(m, si)$ and $\mathbf{finishMission}_i(m, si)$), and setting the goal state of a goal in a scheme instance ($\mathbf{setGoalState}_i(\textit{state}, g, si)$). Communication actions have the form $\mathbf{send}_i(j, \phi)$, where j is the agent name of the receiving agent, and ϕ is the content of the message, which we do not further detail in this paper.

The semantics of the execution of MOISE$^+$ organizational actions, i.e., the definition of when they can be executed (preconditions) and how they affect the organizational entity (postconditions), is specified in [11]. To represent how the execution of organizational actions affects the organizational entity we use the function $\mathcal{O}$ that takes an organizational action with instantiated parameters oa, the agent i that executed the action, and an organizational entity *OE* and yields the updated *OE*. We define $\mathcal{O}(oa, i, OE) = OE'$ to yield the updated organizational entity *OE'* as specified in [11] if the precondition of oa executed by i holds, and otherwise the function is undefined.

We define the semantics of the execution of a MOISE$^+$ MAS as usual as the interleaved execution of the agents participating in the MAS, defined by means of a transition system [17] (see, e.g., [4] for a similar formalization).[5] A transition system for a MAS consists of a set of axioms and transition rules for deriving transitions for the MAS. A transition is a transformation of one state of the MAS into another and it corresponds to a single computation step. Below, we present the transition rule for the execution of organizational actions, where $a_i \xrightarrow{oa} a_i'$ expresses that agent i executes organizational action oa, through which the agent's state changes from a_i to a_i'.

$$\frac{a_i \xrightarrow{oa} a_i' \qquad \mathcal{O}(oa, i, OE) = OE'}{\langle OE, \xi, a_0, \ldots, a_i, \ldots, a_n, \_\rangle \to \langle OE', \xi, a_0, \ldots, a_i', \ldots, a_n, oa_i\rangle}$$

Similar transition rules can be defined for the execution of environment actions and internal actions, but we omit them here since it is not the focus of this paper.

The semantics for sending of messages is as follows, where a_j' represents the state of agent j after having received the message.[6]

$$\frac{a_i \xrightarrow{\mathbf{send}(j,\phi)} a_i'}{\begin{array}{l}\langle OE, \xi, a_0, \ldots, a_i, \ldots, a_j, \ldots, a_n, \_\rangle \to \\ \qquad \langle OE, \xi, a_0, \ldots, a_i', \ldots, a_j', \ldots, a_n, \mathbf{send}_i(j, \phi)\rangle\end{array}}$$

The execution of a MOISE$^+$ MAS results in a computation. We define a computation as a sequence of MAS states, such that each state (except the initial state) can be obtained from the previous by applying a transition rule.

3.2 Linear Temporal Logic

[4]MOISE$^+$ considers communication actions to be organizational actions, but we separate them since their semantics are defined differently.

[5]This implies that only one action is executed at a time. This can be generalized, but it is not needed for the purpose of this paper.

[6]This semantics assumes agent j is always able to receive a message. This is done for simplicity, but it is not essential for our framework.

We use linear temporal logic (LTL) [7] for formalizing organizational constraints. The atomic LTL formulas in our case are "organizational predicates" to express properties of the organizational entity of a MAS state ($\mathcal{L}_{org}$) and the formula of the form $done_i(\mathsf{act})$ to refer to the action that was executed to reach a MAS state. The language $\mathcal{L}_{org}$ is introduced step by step in the sequel. Our LTL language with typical element φ, is then defined by:

$$\begin{array}{lcl} \chi & ::= & \text{any element from } \mathcal{L}_{org} \cup \{done_i(\mathsf{act})\} \\ \varphi & ::= & \chi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \;\mathsf{until}\; \varphi \end{array}$$

The semantics of LTL formulas is as usual defined on traces, which are infinite sequences of states. In our case traces are basically MAS computations. Since LTL is defined on infinite traces, we augment each finite MAS computation t where $s_k = \langle OE, \xi, a_0, \ldots, a_n, \mathsf{act}_i \rangle$ is the final state of t by an infinite sequence of states $s_{k+1}, s_{k+2}, \ldots$ where each s_m with $m > k$ equals $\langle OE, \xi, a_0, \ldots, a_n, \epsilon \rangle$. The semantics of the temporal operators is as usual [7]. The eventuality operator $\Diamond\varphi$ is introduced as an abbreviation for true until φ and its dual $\Box\varphi$ is defined as $\neg\Diamond\neg\varphi$. We also introduce the φ before ψ operator to express that φ occurs before ψ. This operator does not introduce additional expressivity and can be defined in terms of until by $\neg(\neg\varphi \;\mathsf{until}\; \psi)$.

The formula $done_i(\mathsf{act})$ and predicates from $\mathcal{L}_{org}$ form the atoms of the LTL language and are consequently state formulas, i.e., their semantics is defined on MAS states. The semantics of $done_i(\mathsf{act})$ as evaluated on a MAS state is as follows: $\langle OE, \xi, a_0, \ldots, a_n, \mathsf{act}_i \rangle \models done_i(\mathsf{act})$. Since formulas from $\mathcal{L}_{org}$ refer to the organizational entity of a MAS, we will in the sequel often say that an organizational entity satisfies an organizational predicate, which should be taken to mean that the MAS state containing the organizational entity satisfies the predicate.

4. HARD CONSTRAINTS

When an agent enters a MOISE$^+$ MAS, it is not completely free to do what it wants. MOISE$^+$ imposes certain organizational constraints on the participating agents, as already mentioned in Section 1. MOISE$^+$ distinguishes between *hard constraints* and *soft constraints*. Hard constraints "must be enforced to maintain the organizational entity in a consistent state" and "since these constraints cannot be violated by any agent, they should be implemented in the middleware" [14]. Soft constraints, on the other hand, "are related to the deontic dimension and are not guaranteed by the middleware, since the agents are supposed to autonomously decide whether to follow them or not" [14].

In this section, we analyze informally which hard constraints are imposed by MOISE$^+$ (Section 4.1) and we analyze them formally by expressing them in LTL (Section 4.2). We address soft constraints in Section 5. Since MOISE$^+$ does not formally specify when an agent can be said to adhere to organizational constraints, the formalizations provided in this and the next section necessarily form our *interpretation* of the informal definitions of constraints.

4.1 Informal Analysis

In [14], the following hard constraints are distinguished: (i) the number of role players in a group cannot exceed the maximum as specified in the cardinality relation between role and group,[7] (ii) the role compatibility relation should be respected, (iii) an agent can only commit to permitted or obliged missions, (iv) the acquaintance and communication links should be respected, and (v) group and scheme instances can only be created from schemes and groups as specified in the organizational specification.

[7]We have omitted these cardinality constraints from the specification in Section 2.2.

Analyzing these constraints, we identify that all except constraint (iv) concern *preconditions for the execution of organizational actions*. For example, rephrasing (ii) it says that an agent cannot adopt a role if it violates role compatibility. However, these hard constraints do not include all preconditions that are specified in [11]. For example, in [11] a precondition for the organizational action $\mathbf{removeRole}_i(\rho, gi)$ is that the agent i is playing role ρ in group instance gi. It is not clear why certain preconditions were omitted from the hard constraints in [14]. However, since we do not see a reason for including certain preconditions and omitting others, we propose to consider all preconditions for organizational actions as hard constraints.

We thus suggest the following hard constraints:

1. *Preconditions for the execution of organizational actions* should be respected.

2. *Acquaintance and communication links* of the structural specification should be respected.

Informally, we mean by respecting preconditions of organizational actions that an agent can only execute organizational actions if their preconditions hold. Respecting acquaintance links means that an agent i is allowed to have a representation of j iff i is linked through an acquaintance link to agent j [13]. Similarly, respecting communication links means that an agent i can only send a message to another agent j if i is connected through a communication link to j [13].

4.2 Formal Analysis

4.2.1 Preconditions

In this section we formally analyze the hard constraint of respecting preconditions of organizational actions. The first question that needs to be answered is what preconditions are. In [11], all preconditions refer to properties of an organizational entity. Preconditions can thus be expressed using the language $\mathcal{L}_{org}$.

We have informally interpreted the hard constraint as meaning that an agent can only execute an organizational action if its precondition holds. In order to formalize this, we use oa_i to denote an organizational action executed by agent i and $pre(oa_i)$ to denote its precondition expressed in $\mathcal{L}_{org}$ and corresponding to the specification in [11]. Then we formalize the hard constraint as follows, which says that if an agent i has just executed an organizational action oa, its precondition $pre(oa_i)$ should hold in the state in which it was executed.

DEFINITION 1. *(hard constraint: respect preconditions)*

$$\Box\big(\bigcirc(done_i(oa)) \rightarrow pre(oa_i)\big)$$

MOISE$^+$ requires that hard constraints cannot be violated by agents. In practice, this means that the organizatonal middleware should ensure this. In our formal framework, the semantics of the MOISE$^+$ organization should ensure that the hard constraint can never be violated, like the middleware does in the implementation. Formally, this means that the hard constraint should be a validity of the framework. This is in fact the case, as expressed in the following proposition.

PROPOSITION 1. *The hard constraint of Definition 1 is a validity of the semantic framework of Section 3.1.*

This hard constraint expresses required agent behavior *from the perspective of the organization*. However, it does not specify how this should influence the agent's internal reasoning processes. For example, since the agent cannot execute organizational actions if their preconditions do not hold, it seems it would be useful if the agent is aware of this and does not consider those actions in its action selection process. This would also require the agent to have enough information about the organization to determine whether preconditions of organizational actions hold. How the organizational constraints influence an agent's reasoning processes is one of the main issues that we will investigate in future research. Making precise what the organizational constraints are as we do in this paper, is an important and necessary step towards this goal.

4.2.2 Acquaintance and Communication Links

Respecting acquaintance links informally means that an agent i is allowed to have a representation of j iff i is linked through an acquaintance link to agent j [13]. In order to formalize this constraint, we have to clarify two issues: (i) what does it mean that an agent has a representation of another agent, and (ii) what does it mean that an agent has an acquaintance link to another agent?

We interpret (i) as meaning that information about the existence of an agent j can only be made available from the organizational entity to i if i has an acquaintance link to j. However, the formal tools needed to formalize this interpretation of the constraint require a more elaborate formal framework for the interaction of agents with the organization than presented in this paper. Though of great relevance, defining such a framework is beyond the scope of this paper and remains for future work.

We now proceed by formalizing the constraint that an agent i can only send a message to another agent j if i is connected through a communication link to j. In order to formalize this, we have to specify what it means that agents are connected through a communication link. Following [14], we define that an agent i is connected to another agent j through a communication link iff i plays a role ρ in a group instance gi, ρ is connected in gr through a communication link to another role ρ', and j plays role ρ' in the same group instance gi.[8]

Formally, we introduce a predicate $\mathit{hasLink}_{com}(i,j) \in \mathcal{L}_{org}$ to specify that agents i and j are connected through a communication link in a MAS state, defined as follows. Let $i, j \in \mathcal{N}$. We define an auxiliary function $\mathit{r2r}^{OE}_{com} : \mathcal{R} \to \mathcal{P}(\mathcal{R})$ that takes a role and yields the roles that are connected to it through a communication link: $\mathit{r2r}^{OE}_{com}(\rho) = \{\rho' \in \mathcal{R} \mid (\rho, \rho', com) \in \mathcal{L}\}$. We then define that $\mathit{hasLink}_{com}(i,j)$ holds in a MAS state with organizational entity OE iff

$$\exists \rho, gi, \rho' : (\rho, gi) \in \mathit{a2r}(i) \text{ and } \rho' \in \mathit{r2r}^{OE}_{com}(\rho) \text{ and } (\rho', gi) \in \mathit{a2r}(j)$$

We now use the predicate $\mathit{hasLink}_{com}(i,j)$ to formally define the hard constraint, which says that if an agent i has just performed a $\mathbf{send}(j,\phi)$ action, $\mathit{hasLink}_{com}(i,j)$ should hold in the state in which the message was sent.

DEFINITION 2. *(hard constraint: sending messages)*

$$\Box\big(\bigcirc(\mathit{done}_i(\mathbf{send}(j,\phi))) \to \mathit{hasLink}_{com}(i,j)\big)$$

This hard constraint is defined analogously to the hard constraint of Definition 1. However, in contrast with the latter, under the semantics of Section 3.1 the constraint of Definition 2 is not a validity of the semantic framework. This is due to the fact that we have defined the semantics of sending of messages independently from the organizational entity. In order to make the constraint of Definition 2 a validity, the semantics has to be adapted by adding the condition $\mathit{hasLink}_{com}(i,j)$ to the premise of the transition rule for sending messages as done below.

$$\frac{a_i \overset{\mathbf{send}(j,\phi)}{\to} a_i' \qquad \mathit{hasLink}_{com}(i,j)}{\begin{array}{l}\langle OE, \xi, a_0, \ldots, a_i, \ldots, a_j, \ldots, a_n, \_\rangle \to \\ \quad \langle OE, \xi, a_0, \ldots, a_i', \ldots, a_j', \ldots, a_n, \mathbf{send}_i(j,\phi)\rangle\end{array}} \tag{1}$$

PROPOSITION 2. *The hard constraint of Definition 2 is a validity of the semantic framework of Section 3.1, where the transition rule for sending messages is replaced by (1).*

This semantics brings sending of messages under the control of the organizational entity. This corresponds to the structure of the middleware as proposed in [14], where the communication infrastructure is part of the organizational middleware. However, there remains the issue of communication between agents that is not controlled by the middleware. In particular, it may be the case that agents can identify each other by communicating outside the organization. This would then provide the means for circumventing all restrictions on communication imposed by the organization.

Concluding this section, we note that the formalizations of the two hard constraints have a similar structure. They specify that if a certain action has been executed, certain conditions should hold in the state in which it was executed. That is, they *constrain action execution*. The fact that they constrain action execution explains why they can naturally be guaranteed by the organizational middleware, thereby forming *hard* constraints. If the actions are executed through the middleware, the middleware can check whether the conditions on their execution hold, and prevent their execution if necessary.

5. SOFT CONSTRAINTS

In contrast with hard constraints, soft constraints cannot naturally be guaranteed by the middleware. Soft constraints do not constrain action execution by specifying conditions under which actions *cannot* be executed as in the case of hard constraints, but rather specify when actions *should* be executed. It is up to the agents to decide whether to comply with this or not, since organizational middleware cannot force agents to do anything. A sanctioning system can be used to enforce soft constraints [8].

MOISE$^+$ distinguishes the following soft constraints [14]:

1. An agent should *commit to missions that it is obliged to fulfill.*
2. An agent should *achieve goals* that it is obliged to achieve.
3. *Authority links* should be respected.

As will become clear in the sequel, while we have proposed a single formalization for each hard constraint, soft constraints leave more room for interpretation. The reason may be that soft constraints have a less direct link to the organizational middleware. Their purpose is to stimulate agents to do something useful, but it is then up to the agents to decide whether to comply. The organizational middleware may come in again to apply sanctions in case soft constraints are violated (which to the best of our knowledge is currently not part of the MOISE$^+$ middleware). Both for developing middleware that detects violations of soft constraints as well as for developing organization-aware agents that comply with (or

[8]For simplicity, we only consider so-called *intra-group* links [14], which means that links only concern agents of the same group instance, and we omit inter-group links.

can decide to violate) soft constraints, however, it is essential that it becomes clear what the soft constraints mean.[9]

In this section we suggest multiple interpretations and propose corresponding formalizations. We do not argue that one interpretation is necessarily better than another. In fact, this may depend on the domain or even on the scheme or scheme instance under consideration. However, the fact that soft constraints may be interpreted in several ways demonstrates the need to be precise to avoid ambiguity.

In this paper, we focus on the formalization of the first two soft constraints (Sections 5.1 and 5.2, respectively). The third soft constraint informally means that if agent i has an authority link to agent j, agent i is allowed to "have authority on [i], i.e., to control [i]" [13]. We consider authority links part of a more elaborate formal framework for organizational interaction, and as such is without the scope of this paper. As noted, although these issues are very relevant for a complete framework, we focus here on the core of a semantic framework for MOISE$^+$ organizations.

5.1 Commit to Obliged Missions

In this section, we investigate how the soft constraint of committing to obliged missions can be interpreted.

5.1.1 Formalizing Obligation

The first question that needs to be answered is what it means that an agent is obliged to fulfill a mission. In [14] a function is defined that yields the obliged missions for an agent, given an organizational entity. A similar declarative definition is as follows. Let $i \in \mathcal{N}$, $si \in \mathcal{SI}$ and $(s, gis, j) \in \mathit{si2i}(si)$. Agent i is obliged to commit to mission m in a scheme instance si iff the state of the root goal (retrieved from scheme s) of si is unsatisfied ($\mathit{g2state}(si, \mathit{root}(s)) = \mathit{unsatisfied}$), the agent i plays a role ρ in a group instance gi ($(\rho, gi) \in \mathit{a2r}(i)$) that is responsible for a scheme instance si ($gi \in gis$), and the role ρ is obliged to m ($\mathit{obligation}(\rho, m) \in \mathit{DS}$).

This definition says that an agent is obliged to commit to mission m in scheme instance si if the root goal of si is unsatisfied. Intuitively, it makes sense that an agent is obliged to commit to mission m in si if si is not yet completely satisfied (or impossible). However, we argue that this definition is too weak. An agent should only be obliged to commit to a mission if *goals of this mission* are still unsatisfied. Intuitively, if the agent has performed its part of the job, it should no longer be obliged to do it. Formally, this condition can be specified as follows: $\exists g \in \mathit{m2g}(m) : \mathit{g2state}(si, g) = \mathit{unsatisfied}$, replacing $\mathit{g2state}(si, \mathit{root}(s)) = \mathit{unsatisfied}$.

We introduce the predicate $\mathit{obliged}_i(m, si) \in \mathcal{L}_{org}$ to refer to this modified version of when an agent i is said to be obliged to commit to mission m in scheme instance si. Note the difference between $\mathit{obligation}(\rho, m)$ and $\mathit{obliged}_i(m, si)$. The former expresses an obligation associated to a role which exists as part of the organizational specification, while the latter expresses an obligation for a specific agent arising at a certain point in time due to the creation of a specific scheme instance.[10]

Adapting the definition of obligation as done above is important not only to comply with intuitions, but also to obtain an appropriate formalization of the soft constraint that an agent should commit to obliged missions. Without the adaptation, and agent could commit to an obliged mission, achieve all the goals of this mission, finish the mission, and still be obliged to commit to the mission because the root goal of the scheme is still unsatisfied (because other agents have not achieved the goals of their missions yet). This is not desirable, because committing to the mission again is not useful if the goals have already been satisfied (or are impossible).

5.1.2 Formalizing the Constraint

Now that we have a definition of when an agent is obliged to commit to a mission, we can start formalizing the soft constraint that an agent should commit to obliged missions. For convenience of representation, we introduce the predicate $\mathit{committed}_i(m, si) \in \mathcal{L}_{org}$ to express that agent i is committed to mission m in scheme instance si, which is satisfied by an organizational entity iff $(m, si) \in \mathit{a2c}(i)$.[11]

A first attempt to formalize the soft constraint may be the following, which specifies that if the agent is obliged to commit to a mission, it should at some point be committed to it.

$$\Box(\mathit{obliged}_i(m, si) \rightarrow \Diamond(\mathit{committed}_i(m, si))) \qquad (2)$$

This formalization suggests that the agent can postpone committing to an obliged mission arbitrarily long. Presumably, it is however not desirable that an agent postpones the fulfillment of obligations indefinitely. This is related to research in deontic logic where it is argued that achievement obligations (where an agent is obliged to achieve something in the future that is not already (necessarily) true now) need a *deadline* condition to express that they should be achieved before the deadline [2].

To the best of our knowledge, MOISE$^+$ does not address deadlines in relation to commitment to missions. Following [2], however, we do argue that obligations, also in the context of MOISE$^+$, need deadlines. In this paper, we do not investigate how to extend MOISE$^+$ with deadlines. Rather, we discuss which deadlines may be considered and formalize them in LTL.

A general formalization expressing that *if agent* i *is obliged to commit to mission* m *in scheme instance* si*, it should be committed to this mission before deadline* d, can be given as in the following definition.

DEFINITION 3. *(soft constraint: commit to obliged missions)*

$$\Box(\mathit{obliged}_i(m, si) \rightarrow (\mathit{committed}_i(m, si) \text{ before } d))$$

We note several properties of the relation between constraint (2) and the constraint of Definition 3.

PROPOSITION 3. *The constraint of Definition 3 implies constraint (2), i.e., the former is stronger. Assume* d *never occurs, i.e.,* $\mathit{obliged}_i(m, si) \rightarrow \Box\neg d$*. It then holds that constraint (2) is equivalent to the constraint of Definition 3.*

We now investigate possible instantiations of the deadline d. First, we observe that in order to be useful, the deadline should be such that it can be easily identified when it has passed. For the organization this is useful to be able to detect violations of the soft constraint, and for the agent it will be easier to base its reasoning on such deadlines (see also [10]). This means that the deadline should at least be a state formula, i.e., a formula without temporal operators. The language $\mathcal{L}_{org}$ can thus be used, and also a language for expressing properties of the shared environment of the MAS.

[9]In [8] this is not clarified. Although sanctions are discussed, this is not done in relation to soft constraints.

[10]This nicely matches the distinction between norms and obligations as made in deontic logic, where norms exist in the agent's social environment independent of agents, while obligations apply to an agent at a particular time and for particular situations originating from the norms holding for the group of agents the agent is part of and holding over the interval of time the agent is in the group [3].

[11]The result of executing the action $\mathbf{commitMission}_i(m, si)$ is that the mapping $\mathit{a2c}$ is updated with $i \mapsto (m, si)$, and executing a $\mathbf{commitMission}$ action is the only way to update $\mathit{a2c}$ [11].

We do not discuss aspects related to the shared environment in this paper, but we do investigate several instantiations of d using $\mathcal{L}_{\text{org}}$.

If an agent is to commit to obliged missions, it should do this before it becomes impossible. A situation when it becomes impossible to commit to a mission in a scheme instance si, is when si has been removed. That is, assuming that new scheme instances that are created have a label that is different from any instances that have been created before. This deadline can be expressed as $d = \neg\texttt{SI}(si)$, where $\texttt{SI}(si) \in \mathcal{L}_{org}$ is a predicate that we introduce to express that si is a scheme instance, i.e., it is satisfied by an organizational entity iff $si \in \mathcal{SI}$. Although this deadline seems to be more restrictive than constraint (2), it actually is not, as expressed in the following proposition.

PROPOSITION 4. *Assume that new scheme instances that are created have a label that is different from any instances that have been created before. Then if* $d = \neg\texttt{SI}(si)$*, the constraint of Definition 3 is equivalent to constraint (2).*

This means that choosing $d = \neg\texttt{SI}(si)$ does not put a stronger constraint on agent behavior. Nevertheless, an agent trying to satisfy the constraint may use in its reasoning that there may occur a situation, induced by other agent's actions, where it can no longer satisfy the constraint. This may be an incentive not to postpone committing to missions arbitrarily long.

We now consider several other deadlines that we do not formalize for reasons of space. The second deadline is the case where the agent stops playing a role ("removes" a role, in MOISE$^+$ terminology). If an agent i has the obligation $\mathit{obliged}_i(m, si)$ that is due to the fact that the agent is playing role ρ, the obligation will cease to exist if the agent stops playing ρ (assuming the agent does play any other roles that give rise to this obligation). There are cases where this is an appropriate deadline. For example, consider a soccer game between teams A and B where team B has just started an attack, and agent i is playing the keeper role in team A. A defense scheme instance is created for team A, in which the keeper should take part. That is, agent i is obliged to commit to relevant missions in this scheme instance. In this situation, it is not desirable that agent i stops playing the keeper role. However, in case of a lecturer who is obliged to teach a certain course next year, it should be allowed that the lecturer changes jobs before committing to teaching the lecture. The third deadline is when the agent should start pursuing a goal of a mission that the agent should commit to. A disadvantage of this kind of "just in time committing" is that other agents don't know if the agent will join in.

Besides strengthening (2) using deadlines, another approach is to refine the soft constraint saying that the agent should commit "as soon as possible" to obliged missions. Since an agent may have multiple obligations at the same time, we cannot require the agent to commit immediately after the obligation arises. However, we can say that the agent cannot execute any other actions until it has committed to its obliged missions. A variant of the third deadline where instead of using a deadline, the agent should commit as soon as possible after it should start the pursuit of goals, can also be considered.

5.2 Achieve Obliged Goals

Committing to obliged missions is of no use unless the agent also pursues the corresponding goals. This is reflected in the second soft constraint, which says that an agent should achieve goals that it is obliged to achieve. In this section, we investigate that constraint.

For this, we first need to clarify what it means that an agent is obliged to achieve goals. We propose to define this similarly to the way this is done in [12]. An agent i is obliged to achieve a goal g in a scheme instance si, denoted as $\mathit{obliged}_i(g, si) \in \mathcal{L}_{org}$, iff there is a mission m in scheme instance si that the agent has committed to ($\mathit{committed}_i(m, si)$), g is an unsatisfied goal of that mission ($g \in \mathit{m2g}(m)$ and $\mathit{g2state}(si, g) = \mathit{unsatisfied}$), and the preconditions for starting to achieve the goal g are satisfied, i.e., the goals preceding g in the scheme s of the scheme instance si ($\mathit{si2i}(si) = (s, gis, a)$) have been achieved. The latter corresponds to the $ready(s, g)$ predicate of [12], which we do not further detail here.

We thus define that an agent must have committed to a mission in a scheme instance in order for the goals of this mission to be obliged. We have decided to define it like this because otherwise the obligation to commit to a mission and the obligation to achieve the goals of the mission would occur simultaneously. This could suggest that the agent can already start pursuing the goals to fulfill the obligation to achieve them, before it has even committed to the corresponding mission. This is not desirable for two reasons. First, the agent cannot change the state of a goal in a scheme instance unless it is committed to the corresponding mission [11]. Consequently, from the perspective of the organization the agent cannot achieve goals unless it has committed to the mission. Second, it will most likely be beneficial to coordination with the other agents in the organization if agents announce that they are committed to a mission before they start pursuing the goals of the mission. For these reasons it is desirable that an agent commits to a mission before pursuing its goals. This is in line with the semantics of $\mathit{obliged}_i(g, si)$ as defined above, since it implies that goals only become obliged after the agent has committed to the corresponding mission.

In order to express the soft constraint, we introduce the predicate $\mathit{goalState}(g, si, state) \in \mathcal{L}_{org}$ to express that the state of goal g in scheme instance si is $state$, where $state \in \{\mathit{unsatisfied}, \mathit{achieved}, \mathit{impossible}\}$. It is satisfied by an organizational entity iff $\mathit{g2state}(si, g) = state$. A first attempt at specifying the soft constraint is now the following.

$$\Box(\mathit{obliged}_i(g, si) \rightarrow \Diamond(\mathit{goalState}(g, si, \mathit{achieved}))) \qquad (3)$$

This formalization is analogous to (2), and a similar discussion regarding the use of deadlines applies. However, in this case we suggest that the most suitable language for expressing deadlines of obliged goals will be a language for expressing properties of the shared environment of the MAS, rather than $\mathcal{L}_{\text{org}}$. This has to do with the fact several organizational actions are no longer possible once the agent has committed to a mission: it cannot finish the mission before the corresponding goals are achieved or impossible, and it cannot remove the role related to this mission before it has finished the mission. In [12], specific kinds of deadlines are added to MOISE$^+$ goals by extending these with a "time-to-fulfill" attribute, indicating how much time an agent has to achieve the goal.

Another observation with respect to (3) is that it might be considered too strong. If the agent comes to realize while pursuing goals of the mission that they are actually impossible to achieve, it will violate (3) by setting their goal state to impossible. One may argue that this is unfair since the constraint would require the agent to achieve impossible goals. A weaker variant that remedies this is the following.

$$\begin{aligned}&\Box(\mathit{obliged}_i(g, si) \rightarrow \\ &\qquad\qquad \Diamond(\mathit{goalState}(g, si, \mathit{achieved}) \vee \\ &\qquad\qquad\qquad textitgoalState(g, si, \mathit{impossible}))) \qquad (4)\end{aligned}$$

When developing agents that are to comply with these constraints, one would need to think about what it means for the agent that it is obliged to achieve a goal. Presumably, it should try pursuing it. This may be represented naturally in agent programming languages that incorporate a notion of goal, such as the GOAL language. However, even in those languages several aspects need to be clarified with respect to this constraint. For example, how should the agent determine when to start pursuing a goal? If the goal has a deadline, this should influence the agent's reasoning with respect to this, and techniques from [10] may be used. Also, the agent may have to reason about goal conflicts [19], since not all of its obliged goals might be achievable before their deadlines. Moreover, if an agent i has an obliged goal that is achieved or declared impossible by another agent, agent i can drop the goal if it was pursuing it. It is these kinds of questions that are important to address in future research on developing organization-aware agents.

6. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed hard and soft constraints imposed by MOISE$^+$. We have done this by defining a semantic framework for MOISE$^+$ MAS and formalizing and analyzing the constraints using LTL. We have formalized two main hard constraints and shown that they are validities of the semantic framework. We have analyzed soft constraints by proposing and discussing several possible formalizations. The fact that we see several possible formalizations of soft constraints demonstrates the need to be precise. Moreover, following work in deontic logic, we have highlighted the need for deadlines to appropriately constrain agent behavior in the context of soft constraints. Through these analyses, we have provided the foundations for the development of organization-aware agents that function in a MOISE$^+$ MAS.

As for future work, we aim to further investigate the formalization of constraints related to acquaintance and authority links. Moreover, the fact that we see several possible formalizations of soft constraints suggests that it may be useful to extend MOISE$^+$ in order to allow to specify which interpretation is taken. The next main steps with respect to the development of languages and techniques for organization-aware agents are to investigate how organizational constraints may influence an agent's reasoning and decision making, and which information the agent needs from the organization to perform this reasoning. We will investigate how this kind of reasoning can be incorporated into existing agent programming languages, and to what extent existing approaches for reasoning about goals and norms in agent programming languages can be used for this (see, e.g., [19, 16]). We will investigate the generalization of these techniques for organization-aware agent development to other kinds of organizations, such as the OperA framework [5]. This may also shed light on the relations between various organizational modeling languages, such as [15].

7. REFERENCES

[1] J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. Engineering open environments with electronic institutions. *Engineering applications of artificial intelligence.*, 18(2):191–204, 2005.

[2] J. Broersen. Strategic deontic temporal logic as a reduction to ATL, with an application to Chisholm's scenario. In L. Goble and J.-J. Meyer, editors, *DEON'06*, volume 4048 of *LNCS*, pages 53–68. Springer, 2006.

[3] J. Broersen. Issues in designing logical models for norm change. In G. Vouros, A. Artikis, K. Stathis, and J. Pitt, editors, *OAMAS'08*, volume 5368 of *LNCS*, pages 1–17, 2009.

[4] M. Dastani, N. Tinnemeier, and J.-J. Ch. Meyer. A programming language for normative multi-agent systems. In V. Dignum, editor, *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global, 2009.

[5] V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, 2004.

[6] V. Dignum, editor. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global, 2009.

[7] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.

[8] B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. Controlling an interactive game with a multi-agent based normative organizational model. In *COIN'06@ECAI*, 2006.

[9] K. V. Hindriks. Programming rational agents in GOAL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Tools and Applications*. Springer, Berlin, 2009.

[10] K. V. Hindriks, W. van der Hoek, and M. B. van Riemsdijk. Agent programming with temporally extended goals. In *AAMAS'09*, pages 137–144. IFAAMAS, 2009.

[11] J. F. Hübner. *Um Modelo de Reorganizacao de Sistemas Multiagentes*. PhD thesis, 2003.

[12] J. F. Hübner, O. Boissier, and R. H. Bordini. Normative programming for organisation management infrastructures. In *(COIN@MALLOW'009)*, 2009.

[13] J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *SBIA'02*, volume 2507 of *LNCS*, pages 118–128. Springer, 2002.

[14] J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *International Journal of AOSE*, 1(3/4):370–395, 2007.

[15] C. M. Jonker, A. Sharpanskykh, J. Treur, and P. Yolum. A framework for formal modeling and analysis of organizations. *Appl. Intell.*, 27(1):49–66, 2007.

[16] F. Meneguzzi and M. Luck. Norm-based behaviour modification in BDI agents. In *AAMAS'09*, pages 177–184, Budapest, 2009.

[17] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[18] M. Sierhuis, C. Jonker, B. v. Riemsdijk, and K. Hindriks. Towards organization aware agent-based simulation. *International Journal of Intelligent Control and Systems*, (Special Issue on Agent Directed Simulation), To appear.

[19] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Goals in conflict: Semantic foundations of goals in agent programming. *JAAMAS*, 18(3):471–500, 2009.

[20] M. B. van Riemsdijk, K. V. Hindriks, and C. M. Jonker. Organization-aware agent programming: A research agenda. In *ESAW'09*, volume 5881 of LNAI, pages 98–112. Springer, 2009.

[21] M. Wooldridge. Semantic Issues in the Verification of Agent Communication Languages. *Autonomous Agents and Multi-Agent Systems*, 3(1):9–31, 2000.