Expert Systems with Applications xxx (2011) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Distributed norm management for multi-agent systems

Wamberto W. Vasconcelos [a,*], Andrés García-Camino [b], Dorian Gaertner [c], Juan A. Rodríguez-Aguilar [b], Pablo Noriega [b]

[a] Department of Computing Science, University of Aberdeen, AB24 3UE, United Kingdom
[b] Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, 08913 Bellaterra, Catalonia, Spain
[c] Department of Computing, Imperial College London, SW7 2AZ, United Kingdom

ARTICLE INFO

Keywords:
Norm
Software agents
Normative conflict
Multi-agent systems

ABSTRACT

Norms explicitly represent prohibitions, permissions and obligations associated with software agents, changing as agents act and interact in pursuit of their goals. Norms provide means of regulating open and heterogeneous multi-agent systems; however, in order to truly reflect the nature of multi-agent systems, norms should be managed in a distributed fashion. A centralized account of norms creates a single point-of-failure and bottlenecks, and as a result fault-tolerance and scalability are jeopardized. The decentralized management of norms is, nevertheless, a challenging issue and we observe a lack of *truly distributed* computational realizations of normative models. To remedy this, we propose *normative structures*, which allow the propagation of changes in the norms associated with agents, as a result of their actions. Due to the dynamic nature of multi-agent systems and the potential concurrency of agents' actions, *conflicts* may arise, whereby an action is simultaneously prohibited and obliged (or prohibited and permitted). We thus present a run-time algorithm to detect and resolve conflicts during the enactment of a multi-agent system, and show how this algorithm can be put to use within a distributed setup.

1. Introduction

A fundamental feature of regulated open multi-agent systems (MAS) in which heterogeneous and autonomous agents interact, is that participating agents are meant to comply with the conventions of the system. Explicit representations of the prohibitions, permissions and obligations associated with software agents – their *normative positions* (also named *deontic states* by some authors) – have been used for modeling such conventions, providing a means to regulate the observable behavior of agents (Dignum, 1999; Wooldridge, 2002). Agents' normative positions change as agents act and interact in pursuit of their goals.

However, norm-compliance, albeit essential, may be difficult to guarantee in practice because agents acquire and discharge obligations, permissions and prohibitions towards other agents in a dynamic fashion. Moreover, even if we assume full norm compliance, this will not necessarily prevent undesirable situations unforeseen at design time. One such undesirable situation is when *normative conflicts* arise, whereby actions are simultaneously prohibited and obliged (or prohibited and permitted) (Cholvy & Cuppens, 1995; Elhag, Breuker, & Brouwer, 2000; Sartor, 1992; von Wright, 1963). Normative conflicts are undesirable because whatever an agent does (or refrains from doing) violates a norm (Vasconcelos, Kollingbaum, & Norman, 2007, 2009). Such scenarios are particularly complex when agents interact in several concurrent activities and obligations, permissions and prohibitions need to propagate dynamically from one activity to another (or to many others).

We address the problem of regulating the operation of open multi-agent systems in which multiple, concurrent/distributed, interrelated activities take place; the activities are where heterogeneous and autonomous software agents interact. In these circumstances, a centralized representation and/or control of normative positions generates bottlenecks and single points-of-failure that compromise scalability and graceful degradation[1] of the multi-agent system's functionalities. The study of computational models that tackle the distributed management of norms is a little-explored issue which our work looks into.

We introduce *normative structures* as a means to observe and manage the evolution of normative positions in each activity and to manage the propagation of normative positions between

* Corresponding author. Tel.: +44 0 1224 272283; fax: +44 0 1224 273422.
E-mail addresses: wvasconcelos@acm.org (W.W. Vasconcelos), andres@garcia-camino.es (A. García-Camino), dg00@doc.ic.ac.uk (D. Gaertner), jar@iiia.csic.es (J.A. Rodríguez-Aguilar), pablo@iiia.csic.es (P. Noriega).

[1] In centralized models we have an "all or nothing" scenario, in which the system will either work if the centralized information repository is accessible, otherwise nothing works. In a decentralized set-up, the information and/or control is distributed so the system will experience a gradual cessation (or *graceful degradation* (Cristian, 1991)) of its functionalities, as less information becomes available.

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.108

distributed activities. We propose an algorithm that exploits the distributed nature of normative structures to deal, at run time, with the emergence of conflicts as agents interact – the algorithm detects and resolves normative conflicts as they arise during the execution of a MAS. We provide guarantees to our algorithm, namely, correctness, termination and worst-case complexity.

To make the problem amenable to analysis, we make the following assumptions:

- We assume that MAS are *open* inasmuch as agents' internal states are inaccessible to the system and agents may enter and leave the MAS as long as they comply with norms. We also assume that agents have the abilities to comply with norms, and will perform "legitimate" actions. We only consider agent actions consisting of speech acts (Searle, 1969) (or actions which may be represented as speech acts, as in, for instance, truthfully informing the concerned parties about actions performed).
- We assume that MAS regulations belong to two types. The first type is embedded in constitutive and procedural norms that are regimented and outside the control of participating agents.[2] The second type corresponds to non-regimented functional norms. Following (Boella, Pigozzi, & Van der Torre, 2009), we do not assume any governance mechanism aside from the regimentation imposed by the MAS itself, thus agents may choose to violate a non-regimented norm. Hence, we do not assume full norm compliance by the agents – we deal with any normative conflicts that may arise not only from violation of a norm but from the normative inconsistencies inherent in the system and brought about by agent interactions. Additionally, we do not assume that there are meta-level or second-level rules that govern the change of existing rules, hence we will not address the problem of norm change as such.
- We make assumptions about the structuring of agent interactions: agents interact in repetitive *activities* regulated by their own particular conventions (procedural and functional). Several of these activities may be simultaneously happening at any given time, with possibly many instances of one same activity. An agent may participate in one activity with other agents and in another activity with possibly different agents. Moreover, an agent may participate simultaneously in more than one activity and may move from one activity to another (or many others). The actions an agent performs in one activity may cause updates in the agents' normative positions, and this may affect the actions agents perform in the future, in that or in other activities.

Some of these assumptions refer to more operational aspects of the problem, and we will elaborate upon them in the rest of the paper. In Sections 6 and 7 we discuss further assumptions and how to relax them. It is worth mentioning that, like most work in this field to a certain extent (and we shall discuss this in more detail in the next section), ours responds to two main drives, namely, (i) the need to deal with the essential deontic aspects of norms and, therefore, their (inferential) consequences on agent behavior; as well as, (ii) the interest in building realistic MAS and hence the need to examine algorithmic questions and computational architectures that may support regulated and open MASs. Our contributions address the second drive.

In the present work we focus on functional norms, expressed in a rule-based formalism that establishes under which conditions agents become or cease to be subject to *normative positions*, that is, obligations, prohibitions and permissions towards specific actions, represented as atomic deontic formulae. We have shown (García-Camino, Rodríguez-Aguilar, Sierra, & Vasconcelos, 2009) that our representation is expressive enough to capture many useful normative concepts in the literature. Although the formalism of (García-Camino et al., 2009) is more sophisticated than the one we use in the present paper, the simplification does not affect our contributions in the present paper; in Section 7 we explain how we deal with richer languages.

The structure of this paper is as follows. In the remainder of this section we introduce a scenario to serve as a running example throughout the article. In Section 2 we present basic concepts and formally define normative structures. In Section 3 we define normative conflicts. Section 4 presents an algorithm for conflict resolution. Section 5 elaborates on an architecture that allows the distributed management of normative positions and solves conflicts via the algorithm presented in Section 4. Finally, we survey related work in Section 6, and draw conclusions and comment on future work in Section 7.

1.1. Scenario

To illustrate and motivate our exposition we use a supply-chain scenario in which companies and individuals come together in an (electronic) marketplace to conduct business. The overall transaction procedure may be organized as six activities, represented as oval nodes in the diagram of Fig. 1. They involve different participants whose behavior is coordinated through protocols (that is, pre-defined sequences of messages with a shared content language). In this scenario agents can play one of four roles: accountant (represented as *acc*), *client*, supplier (represented as *supp*) and warehouse manager (represented as *wm*). The triangles represent *transitions* among activities, that is, "meeting points" where agents synchronize their movements out of an activity[3] and into another activity (or choice of other activities) (Arcos, Esteva, Noriega, Rodríguez-Aguilar, & Sierra, 2005). The lines and arrows connecting activities and transitions indicate the order in which activities can be undertaken.

After registering at the marketplace, clients and suppliers get together in the *negotiation* activity where they agree on the terms of their transaction, *i.e.* prices, amounts of goods to be delivered, deadlines and other details. In the *contract* activity, the order becomes established and an invoice is prepared. The client will then participate in a *payment* activity, verifying his credit-worthiness and instructing his bank to transfer the correct amount of money. The supplier in the meantime will arrange for the goods to be delivered (*e.g.*, via a warehouse manager) in the *delivery* activity. Finally, agents can leave the marketplace conforming to a predetermined *exit* protocol. The marketplace accountant participates in most of the activities as a trusted provider of auditing services. In the remainder of this article we shall build on this scenario to exemplify the notion of normative structure and to illustrate our approach to conflict detection and resolution in a distributed setting.

2. Normative structures

As argued above, we consider that the specification of an open and regulated MAS encompasses the specification of a coordination level and a normative level. This allows the designer to separate concerns. On the one hand, the designer can concentrate on

[2] Those regimented procedural rules are expressed either as protocols that organize procedural prescriptions in each activity, or as conventions that regulate when agents may move from one activity to another.

[3] Activities are also called *scenes* in the electronic institution literature, as in *e.g.*, (Esteva, Vasconcelos, Sierra, & Rodríguez-Aguilar, 2004; García-Camino, Noriega, & Rodríguez-Aguilar, 2007; García-Camino, Rodríguez-Aguilar, Sierra, & Vasconcelos, 2006).

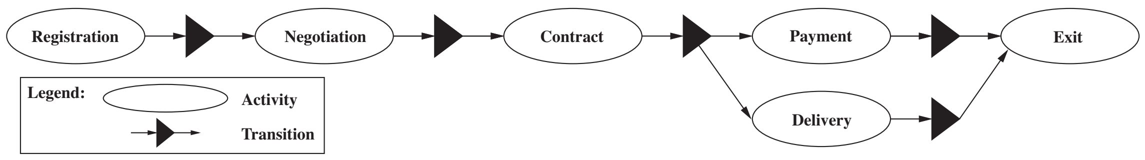

Fig. 1. Activities of a virtual marketplace.

modeling interactions within activities along with the dependencies between activities when specifying the coordination level. As a result, the designer can produce a specification like the one in Fig. 1. Agents in a MAS will then be able to interact according to the protocols of the activities specified at the coordination level; such activities can be naturally enacted in a distributed fashion. As an activity evolves, normative positions may change due to agents' actions. For instance, when a seller agrees on a contract, it is expected to deliver certain goods. Furthermore, actions in one activity may have an effect on the enactment of other activities. Using the running example, one can consider negotiation, contract-signing and delivery as different (but related) activities. A contract may be created during the negotiation activity, it may be agreed during the contract-signing activity, and it may be (partially) fulfilled during the delivery activity. Therefore, the specification of the triggering and removal of normative positions becomes essential at the normative level. Hereafter, we focus on offering the MAS designer a formalism that eases the specification of the activation and removal of normative positions.

Actions and/or normative positions in some activity cause changes in the normative positions of agents in other activities. We are concerned with the propagation of normative positions within a network of distributed activities as a consequence of agents' actions. We refer to the normative state of an activity as the set of normative positions that are in force at a particular moment during the enactment of the activity. Furthermore, a normative transition defines how a change in a normative state affects the normative state of other activities. As for effects of actions, a normative structure includes both of these notions (normative states and normative transitions) to establish the existing relationships among different activities. In order to define our norm language and specify how normative positions are propagated, we have been inspired by multi-context systems (Giunchiglia & Serafini, 1994).

In this section, we present a simple language capturing these aspects and introduce the notions of normative states, normative transition and normative structure formally. We give the intended semantics of these concepts and show an example of how a MAS can be regulated via normative positions.

2.1. Basic concepts

In this subsection we introduce the basic concepts needed to define a normative structure. The building blocks of our language are *terms* and *atomic formulae*:

Definition 1. A term, denoted generically as τ, is a constant, a variable or a functional expression $f^n(\tau_0,\ldots,\tau_n)$, where f is an n-ary function symbol and all τ_i, for $0 \leqslant i \leqslant n$, are terms.

Constants, generically denoted as a, b, c (possibly with subscripts), are strings starting with a lowercase letter, with or without subscripts, as in, *e.g.*, $seller_3$, *copper*. Variables, generically denoted as v, w, x, y, z (possibly with subscripts), are strings starting with an uppercase letter, with or without subscripts, as in, *e.g.*, $Role_2$, *Price*. We build atomic formulae using terms as parameters:

Definition 2. An atomic formula is any construct $p^m(\tau_0,\ldots,\tau_m)$, where p^m is an m-ary predicate symbol and all τ_i, for $0 \leqslant i \leqslant m$, are terms.

We make use of first-order *substitutions* (Apt, 1997; Fitting, 1990), formally defined as:

Definition 3. A substitution $\sigma = \{x_0/\tau_0,\ldots,x_n/\tau_n\}$ is a finite and possibly empty set of pairs x_i/τ_i, $0 \leqslant i \leqslant n$, x_i being a variable and τ_i a term as defined above.

We define the application of a substitution in accordance with (Fitting, 1990) – a substitution σ is a *unifier* of two terms τ_1, τ_2, iff $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$, "$\cdot$" being defined as:

Definition 4. The application of a substitution σ to a term or atomic formula is as follows:

1. $c \cdot \sigma = c$ for a constant c;
2. $x \cdot \sigma = \tau \cdot \sigma$ if $x/\tau \in \sigma$; otherwise $x \cdot \sigma = x$;
3. $f^n(\tau_0,\ldots,\tau_n) \cdot \sigma = f^n(\tau_0 \cdot \sigma,\ldots,\tau_n \cdot \sigma)$.
4. $p^m(\tau_0,\ldots,\tau_m) \cdot \sigma = p^m(\tau_0 \cdot \sigma,\ldots,\tau_m \cdot \sigma)$.

We focus on an expressive class of MASs in which interaction is carried out by means of illocutionary speech acts exchanged among participating agents. We initially define *illocution schemata*:

Definition 5. An illocution schema I is any atomic formula $p(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6)$ where

- p is an element of a set of illocutionary particles (*e.g.*, *inform*, *request*, etc.);
- τ_1, τ_3 are variables or agent identifiers;
- τ_2, τ_4 are variables or role identifiers;
- τ_5 is a variable or an arbitrary term representing the contents of a message, built from a shared language;
- τ_6 is a variable or a $t \in \mathbb{N}$ representing a time stamp.

The intuitive meaning of $p(Ag, R, Ag', R', m, t)$ is that agent Ag playing role R has sent message m to agent Ag' playing role R' at time t.[4] We differentiate a subclass of illocution schemata which do not contain any variables, that is, they are ground atomic formulae. We name this subclass the *illocutions* and these are generically denoted as $\bar{\mathsf{I}}$. An example of an illocution is *inform*(ag_4, *supp*, ag_3, *client*, *offer*(*wire*, 12), 10). Illocution schemata are useful in the description of a protocol, when the precise values of a message to be exchanged should be left unspecified. During the enactment of the protocol, however, agents will produce the actual values which will give rise to a ground illocution.

We now define normative positions:

Definition 6. Normative positions, generically referred to as N, are of the form *per*(I), *prh*(I) or *obl*(I), representing, respectively, a permission, a prohibition or an obligation on an illocution schema I.

Normative position *per*($p(Ag, R, Ag', R', m, t)$) intuitively means that agent Ag playing role R is *permitted* to send message m to agent Ag' playing role R' until time t. A normative position *prh*($p(Ag, R, Ag', R', m, t)$) means that agent Ag playing role R is *prohibited* from

[4] The time stamp of illocution schema can be obtained in various ways, with varying degrees of sophistication, accuracy and usefulness. We discuss these in Section 7.

sending message m to agent Ag' playing role R' until time t; similarly, $obl(p(Ag,R,Ag',R',m,t))$ means that agent Ag playing role R is *obliged* to send message m to agent Ag' playing role R' until time t.

2.2. Normative states, transitions and structures

Using the concepts introduced previously, we now provide formal definitions for normative states, transitions and structures. We first define normative state as follows:

Definition 7. A normative state Δ_s of an activity s is a finite set of pairs $\langle \bar{\mathsf{I}}, t\rangle$ (illocutions) or $\langle \mathsf{N}, t\rangle$ (normative positions), $t \in \mathbb{N}$, representing, respectively, that $\bar{\mathsf{I}}$ was uttered at instant t and normative position N was established at t.

Normative states record both the events (what was uttered and when) and the normative positions occurring in an activity. In a pair $\langle p(ag,r,ag',r',\tau,t'),t\rangle \in \Delta$, it is always the case that $t = t'$, that is, the time the illocution was uttered and their time as recorded in Δ are always the same.[5] In a pair $\langle obl(p(ag,r,ag',r',\tau,t')),t\rangle \in \Delta$, t may be different from t', as t' is time by when the illocution ought to be uttered and t is the time when the normative position was generated; however, $t' \geqslant t$, that is, the time t' of the illocution must be at or after the time of the norm establishment t. The same applies to *per* and *prh*.

A sample normative state of the *delivery* activity of our scenario (referred as *dlvry*), denoted as Δ_{dlvry}, is:

$$\left\{\begin{array}{c} \langle inform(bob, seller, kev, wm, deliver(wire, 200), 20), 20\rangle \\ \langle obl(inform(kev, wm, bob, client, deliver(copper, 50), 30)), 20\rangle \\ \langle prh(inform(kev, wm, ann, client, deliver(iron, 25), 30)), 20\rangle \end{array}\right\}$$

In the normative state above, agent *bob* taking up the *seller* role has delivered 200 kg of wire at time 20 to agent *kev* taking up the warehouse manager role *wm*.[6] Agent *kev* has an obligation (established at time 20) to deliver 50 kg of copper to *bib* by time 30; *kev* is furthermore prohibited from delivering 25 kg of iron to *ann*, another *client*, until time 30.

The meaning of normative positions in a normative state is as follows: a pair $\langle per(inform(ag,r,ag',r',\tau,T),T'\rangle \in \Delta$ means that agent *ag* under role *r* is permitted to send an *inform* message with contents τ to agent ag' under role r' from the time T' when the normative position was established up until the time stamp T, $T' \leqslant T$. Similarly, $\langle obl(inform(ag,r,ag',r',\tau,T),T'\rangle \in \Delta$ means that *ag* (under role *r*) is obliged to send the *inform* message to ag' between the time T' of the norm establishment until T, and $\langle prh(inform(ag,r,ag',r',\tau,T),T'\rangle \in \Delta$ means that *ag* (under role *r*) is prohibited from sending the *inform* message to ag' between T' and T.

A more useful way to capture the meaning of normative positions is by describing *norm violations*, that is, those circumstances when one can infer that an agent behaved in a non-norm compliant fashion (Aldewereld, Vázquez-Salceda, Dignum, & Meyer, 2005). Given a normative state Δ and the current time t_{now} we can define the conditions for the violation of an obligation as "*if there is an obligation of the form* $\langle obl(p(\tau_1,\tau_2,\tau_3,\tau_4,\tau_5,t_u)),t_e\rangle \in \Delta$, $t_e \leqslant t_u \leqslant t_{now}$, *and there is no illocution of the form* $\langle p(\tau'_1,\tau'_2,\tau'_3,\tau'_4,\tau'_5,t'_u),t'_u\rangle \in \Delta$, *such that* $p(\tau_1,\tau_2,\tau_3,\tau_4,\tau_5)\cdot\sigma = p(\tau'_1,\tau'_2,\tau'_3,\tau'_4,\tau'_5)\cdot\sigma$, *for some substitution* σ, *and* $t'_u \leqslant t_u$, *then a violation of the obligation has occurred – the obliged utterance was not uttered by* t_u, *the obligation 'deadline'*". Similarly, we address a prohibition violation: we specify the conditions establishing when the prohibited utterance was proffered within the prohibition 'period'. One way to define the violation of permissions is via the description of those situations when an utterance which is not explicitly permitted has been proffered; this is rather strict, but it may be required in highly regulated agent encounters. Alternative formulations can be given to the norm violations above, such as the ones defined in (García-Camino et al., 2009).

Normative states evolve over time: as agents interact, their utterances are recorded; these utterances may cause normative positions to appear or disappear. Activities are connected to one another via normative transitions that specify how utterances and normative positions in one activity (represented in its normative state) affect other activities. Illocutions uttered in one activity may have an effect in other activities. Normative transitions define the conditions under which a normative position is updated. These conditions are either utterances and/or norms associated with a given activity, which cause the addition or removal of a normative position, possibly in a distinct activity. We capture dynamic aspects of norms (and their relationships across distinct activities) via normative transitions:

Definition 8. A normative transition NT is defined as:

$$\mathsf{NT} ::= \mathsf{LH} \Rrightarrow \mathsf{RH}$$
$$\mathsf{LH} ::= (id : \mathsf{D}) | \mathsf{LH}, \mathsf{LH}$$
$$\mathsf{D} ::= \mathsf{N} | \mathsf{I}$$
$$\mathsf{N} ::= per(\mathsf{I}) | prh(\mathsf{I}) | obl(\mathsf{I})$$
$$\mathsf{RH} ::= \oplus(id : \mathsf{N}) | \ominus (id : \mathsf{N})$$

where I is an illocution schema, N is a normative position, *id* is the identifier of an activity and RH is the addition or removal of a normative position.

We endow our language of Definition 8 above with the usual operational semantics of rule-based languages, as formalized in (García-Camino et al., 2009), and which we informally describe below. Our language adds (via operator "$\oplus$") or removes (via operator "$\ominus$") normative positions to/from normative states, capturing dynamic aspects of norms: as agents interact, normative positions are assigned and revoked (that is, removed). For instance, if an agent bids in an auction, then an obligation is put in place for the agent to pay for the item; agents that have not registered with a third party to obtain a letter of credit are forbidden to bid; an obligation to pay is removed if the agent to whom the obligation refers performs the payment. We show in Fig. 2 sample normative transitions. Normative transition nt_0 is not part of our running example, but we included it to illustrate the situation when an obligation is in place and it is fulfilled – the obligation is then removed from the normative state. We discuss the other examples in Section 2.3. Based on the notion of normative transition, we can offer a straightforward definition of *normative structure*:

Definition 9. A normative structure is a pair $\langle S, NT\rangle$, where S is a finite set of normative states and NT is a finite set of normative transitions.

In other words, specifying a normative structure amounts to specifying a rule-based program (a collection of rules) that makes explicit how normative positions flow between the normative states of activities and how they are removed as agents interact. Therefore, the triggering of normative transitions changes normative states by adding and removing normative positions.

Here we give the intuitive semantics of normative transitions by describing how they change normative states. Given a normative transition, we refer to the activities on the left-hand side of the rule as incoming activities and the activities on the right-hand side of the rule as outgoing activities. Each transition is triggered once for each substitution that unifies the left-hand side LH of

[5] Hereafter we drop the index of a normative state that refers to its activity unless it is required to distinguish between normative states of different activities.

[6] Following the terminology from (Searle, 1995) and the computational model of an activity from (García-Camino et al., 2009), we design our MAS in order to incorporate that informing about a delivery "counts as" a factual delivery.

$$nt_0: \left(\begin{array}{l}(pmnt: obl(inform(Ag_1,R_1,Ag_2,R_2,M,T))),\\(pmnt: inform(Ag_1,R_1,Ag_2,R_2,M,T'))\end{array}\right) \Rrightarrow \ominus (pmnt: obl(inform(Ag_1,R_1,Ag_2,R_2,M,T)))$$

$$nt_1: (ngtn: request(B, buyer, S, seller, sell(It,A,P),T_1)) \Rrightarrow \oplus(ngtn: per(accept(S, seller, B, buyer, sell(It,A,P),T_2)))$$

$$nt_2: \left(\begin{array}{l}(ngtn: accept(S, seller, B, buyer, sell(It,A,P),T_1)),\\(pmnt: inform(B, buyer, S, seller, pay(P),T_2))\end{array}\right) \Rrightarrow \oplus(dlvry: obl(inform(S, seller, B, buyer, deliver(It,A),T_3)))$$

$$nt_3: (dlvry: obl(inform(s_1, seller, B, buyer, deliver(copper,A),T_1))) \Rrightarrow \oplus(dstr: obl(inform(d_1, distrib, s_1, seller, deliver(copper,A),T_2)))$$

$$nt_4: (dstr: obl(inform(d_1, distrib, S, seller, deliver(copper,A),T_1))) \Rrightarrow \oplus (mnfc: obl(inform(m_1, M, d_1, distrib, deliver(copper,A),T_2)))$$

Fig. 2. Sample normative transitions.

the transition with a set of elements from possibly several normative states of its incoming activities. An utterance or a normative position on the left-hand side of a transition holds if, and only if, it unifies with an utterance or normative position appearing in the normative state of an incoming activity. Every time a transition is triggered, the update specified on the right-hand side of that transition is carried out, thus adding to or removing from the normative state of an activity a normative position.

Conflicts may arise after the addition of normative positions: if a conflict arises, we make use of an algorithm to decide whether to ignore the new normative position or to "adjust" old normative positions to avoid conflicts. We explain this in more detail in Sections 3 and 4.

2.3. Example

We now discuss normative transitions $nt_1,\ldots,nt_4$ of Fig. 2 for our scenario of Section 1.1. Normative transition nt_1 illustrates how one single normative state can be modified; nt_2 makes reference to more than one normative state in its left-hand side. Finally, nt_3 and nt_4 illustrate a "normative flow" whereby the appearance of a new obligation in one normative state generates other obligations in different normative states.

In our scenario, during the *negotiation* activity (represented as *ngtn*), a request of a buyer *B* to a seller *S* to sell an amount *A* of item *It* at price *P* leads to the introduction of a normative position in the same normative state. The normative position is a permission for *S* to accept the request of *B*. This is formalized by normative transition nt_1. Notice that the request to sell issued by buyer *B* within the negotiation activity would change the negotiation normative state, allowing the firing of nt_1.

In the *negotiation* activity (*ngtn*), if a seller *S* accepts the offer of a buyer *B* to sell an amount *A* of item *It* at price *P* and furthermore, if in the *payment* activity (represented as *pmnt*) the same buyer has paid *P* to seller *S*, then that introduces to the *delivery* activity an obligation on the seller agent to deliver the sold item *It*.[7] This is captured by normative transition nt_2. The illocutions uttered within the negotiation and payment activities would change the negotiation normative state and payment normative state respectively, triggering the obligation in the delivery normative state.

During the *delivery* activity (*dlvry*), the creation of an obligation on seller s_1 to deliver copper leads to the propagation to the *distribution* activity (*dstr*) of an obligation on distributor d_1 to deliver that amount of copper to s_1. We represent thisas nt_3. Finally, during the *distribution* activity, the creation of an obligation on the distributor d_1 to deliver some copper allows the propagation to the *manufacture* activity (*mnfc*) of an obligation on m_1 playing a manufacturer role *M* to deliver that amount of copper to d_1. This is captured by nt_4.

[7] We assume that this payment was made for item *It*. An appropriate labeling during the procurement process is needed but omitted here for simplicity.

3. Normative conflicts

In this article, normative positions refer to illocutions (as opposed to arbitrary actions). Normative conflicts arise when an illocution is simultaneously obliged and prohibited. The terms deontic *conflict* and deontic *inconsistency* have been used interchangeably in the literature to refer to situations in which actions are simultaneously associated with different modalities (von Wright, 1963). However, in this article we adopt the view of (Elhag et al., 2000) in which the authors suggest that a deontic inconsistency arises when an action is simultaneously permitted and prohibited – since a permission may not be acted upon, no conflict actually occurs. The situation when an action is simultaneously obliged and prohibited is, however, a normative conflict, as both obligations and prohibitions influence agents' behaviors in a conflicting fashion.

We use *unification* of first-order terms (Apt, 1997; Fitting, 1990) as a means to detect and resolve conflicts between normative positions. Unification is a fundamental problem in automated theorem proving and many algorithms have been proposed (Fitting, 1990). Unification is based on the concept of substitution (*viz.* Def. 3 and 4), that is, the set of values for variables in a computation. We shall use unification in the following way:

Definition 10. $unify(\tau_1,\tau_2,\sigma)$ holds iff $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$, for some σ; $unify(p^n(\tau_0,\ldots,\tau_n),\ p^n(\tau'_0,\ldots,\tau'_n),\sigma)$ holds iff $unify(\tau_i,\tau'_i,\sigma)$ holds, $0 \leqslant i \leqslant n$, and some σ.

We assume that *unify* is based on a suitable implementation of a unification algorithm that (i) always terminates (possibly failing, if a unifier cannot be found), (ii) is correct and (iii) is of linear computational complexity. The *unify* relationship checks, on the one hand, that substitution σ is a unifier, but can also be used to find σ. By extending the definition of *unify* for handling normative positions, we can use unification for detecting a conflict between two normative positions:

Definition 11. A conflict arises between two normative positions N and N' under a substitution σ, denoted as $\mathbf{conflict}(\mathsf{N},\mathsf{N}',\sigma)$, if and only if $\mathsf{N} = prh(\mathsf{I})$, $\mathsf{N}' = obl(\mathsf{I}')$ and $unify(\mathsf{I},\mathsf{I}',\sigma)$.

That is, a prohibition and an obligation are in conflict if, and only if, their illocutions unify under σ. The substitution σ, called here the *conflict set*, unifies the agents, roles and atomic formulae. For instance, an obligation to every agent A_1 enacting any role at any time to inform *deliver*(*copper*,*X*) to any agent A_2 enacting any role at any time, represented as

$$obl(inform(A_1, R_1, A_2, R_2, deliver(copper, X), T))$$

and a prohibition on an specific agent a_1 enacting the specific role r_1 to inform *deliver*(*Y*, *dest*) to another specific agent a_2 enacting specifically role r_2 in any time T', represented as

$$prh(inform(a_1, r_1, a_2, r_2, deliver(Y, dest), T'))$$

are in conflict as their illocutions unify under

$$\sigma = \{A_1/a_1, R_1/r_1, A_2/a_2, R_2/r_2, Y/copper, X/dest, T/T'\}$$

Agent a_1 under role r_1 is simultaneously obliged and forbidden to deliver *copper* to agent a_2 under role r_2 at a particular destination *dest*.

Inconsistencies caused by the same illocution being simultaneously permitted and prohibited can be formalized similarly. In this article we focus on prohibition/obligation conflicts, but the computational machinery introduced in Section 4 can equally be used to detect prohibition/permission inconsistencies, if we replace *obl* with *per*.

4. Conflict resolution

Once a conflict (as defined in Section 3) has been detected, we propose to employ the unifier to *resolve the conflict*. By conflict resolution we mean the careful manipulation of normative positions aiming at restricting the value their variables may have, thus eliminating any conflicts. In our approach, normative positions are *annotated* with substitutions representing values their variables *must not* have, in order to avoid any conflicts with other normative positions.

In our example of Section 3, if the variables in *prh*(*inform*(a_1, r_1,a_2,r_2,*deliver*(*Y*,*dest*),*T'*)) *do not* get the values specified in substitution $\sigma = \{A_1/a_1,R_1/r_1,A_2/a_2,R_2/r_2,Y/copper,X/dest,T/T'\}$) then there will be no conflicts, that is, there are no conflicts as long as the prohibition does not address the delivery of copper (the other pairs of unifications do not affect the prohibition, though). However, rather than computing the complement set of a substitution (that is, all possible values for variables which do not unify – these can be an infinite set) we propose to annotate the prohibition with the unifier itself and use it to determine what the variables of that prohibition *cannot* be in future unifications in order to avoid a conflict. We therefore denote annotated prohibitions as $prh(\bar{\mathsf{I}}) \odot \Sigma$, where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, is a set of unifiers. Annotated normative positions are interpreted as deontic constructs with *curtailed* influences (Vasconcelos, Kollingbaum, & Norman, 2009), that is, their effect (on agents, roles and illocutions) has been limited by the set Σ of unifiers. Permissions and obligations do not get annotations: they remain in their original format.

A prohibition may be in conflict with various obligations in a given normative state Δ and we need to record (and possibly avoid) all these conflicts. We define below an algorithm which ensures that a normative position will be added to a normative state in such a way that it will not cause any conflicts. We propose a fine-grained way of resolving normative conflicts via unification. We detect the overlapping of the normative positions' influences, *i.e.* how they affect the behavior of the concerned agents, and we curtail the influence of the normative position by appropriately using the annotations when checking if the normative position applies to illocutions. Algorithm 1 depicts how we manage the insertion of a normative position N into a normative state Δ obtaining a set of additions/removals of normative positions Cs. These updates make use of prohibitions annotated with a set $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ of *conflict sets* σ_i indicating which bindings for variables have to be avoided for conflicts not to take place.

Algorithm 1. *manageNorm*

```
Input: $\oplus(s : \mathsf{N})$, $\Delta$
output: $\mathsf{Cs}$
1: $t_c := current\_time()$;
2: case $\mathsf{N}$ of
3:   $(per(\mathsf{I})$: $\mathsf{Cs} := \{\oplus\ (s{:}\mathsf{N})\}$
4:   $(prh(\bar{\mathsf{I}})$: if $\exists\langle \mathsf{N}', t\rangle \in \Delta$, $t \leqslant t_c$, conflict$(\mathsf{N},\mathsf{N}',\sigma)$ then
5:            $\mathsf{Cs}{:=}\emptyset$
6:          else
7:            $\mathsf{Cs}{:=}\{\oplus\ (s : \mathsf{N} \odot \emptyset)\}$
8:   $prh(\mathsf{I})$:
9:     begin
10:        $\Sigma{:=}\emptyset$
11:        for all $\langle \mathsf{N}', t\rangle \in \Delta$, $t \leqslant t_c$, conflict$(\mathsf{N}, \mathsf{N}', \sigma)$ do
12;          $\Sigma := \Sigma \cup \{\sigma\}$
13:        $\mathsf{Cs} := \{\oplus(s : \mathsf{N} \odot \Sigma)\}$
14:     end
15:   $obl(\mathsf{I})$:
16:     begin
17:        $\mathsf{Cs} := \{\oplus(s : \mathsf{N})\}$
18:        for all $\langle \mathsf{N}' \odot \Sigma, t\rangle \in \Delta$, $t \leqslant t_c$, conflict$(\mathsf{N}',\mathsf{N},\sigma)$ do
19:          if $\mathsf{N}' = prh(\bar{\mathsf{I}})$ then
20:            $\mathsf{Cs} := \mathsf{Cs} \cup \{\ominus(s : \mathsf{N}' \odot \Sigma)\}$
21:          else
22:            if $\sigma \notin \Sigma$ then
23:              $\mathsf{Cs} := \mathsf{Cs} \cup \{\ominus\ (s{:}\mathsf{N}' \odot \Sigma),\ \oplus\ (s{:}\mathsf{N}' \odot (\Sigma \cup \{\sigma\}))\}$
24:     end
25: end case
26: return $\mathsf{Cs}$
```

The algorithm uses a **case of** structure to differentiate the kinds of normative positions N to be inserted. Line 3 addresses the case when N is a permission: in this case, the set of commands Cs consists of a single command to add N (since it is a permission, it does not have annotations). Lines 4–7 address the case when we attempt to add a ground prohibition on $\bar{\mathsf{I}}$ to a normative state: if it conflicts with any obligation in Δ established before or at the current time (that is, $t \leqslant t_c$), then it is discarded (Line 5); otherwise Cs will contain a command to insert the new normative position (with an empty annotation) to the appropriate state (Line 7). Lines 8–14 describe the situation when N is a prohibition on a non-ground illocution schema I. In this case, the algorithm initializes an annotation Σ as an empty set and loops (Lines 11–12) through the normative positions N' established before or at the current time (that is, $t \leqslant t_c$) and which are in conflict with N. For each of such normative positions N', the algorithm updates Σ by adding the newly found conflict set σ to it. By looping through Δ, we are able to check any conflicts between the new prohibition and all existing obligations, adequately building the annotation Σ to be used in the command to include the annotated normative position $\mathsf{N} \odot \Sigma$ in Line 13.

Lines 15–24 describe how obligations are dealt with: obligations are always added, hence Cs is initialized to $\{\oplus(s,\mathsf{N})\}$ (Line 17). However, the new obligation may be in conflict with existing prohibitions in Δ, hence the algorithm defines a loop (Lines 18–23) through all annotated prohibitions $\langle \mathsf{N}, t\rangle \in \Delta$, whose activation time t is less than or equal to the current time t_c, and which are in conflict with N, under σ. For each prohibition in this loop, we must check if it concerns a ground illocution (Line 19) – in this case, we add a command in Cs to remove the ground prohibition (its annotation is the empty set). If the prohibition concerns a non-ground illocution, then we must check if the conflict set $\sigma \notin \Sigma$: if $\sigma \in \Sigma$ then the conflict does not really happen as the curtailed prohibition does not overlap with N. If there is indeed a conflict (that is, $\sigma \notin \Sigma$) then we must update the annotation Σ of the prohibition to add the new conflict set σ – we do so by creating two commands in Cs, one to remove the old normative position, *viz.* $\ominus(s : \mathsf{N}' \odot \Sigma)$ and one to add the new updated normative position, *viz.* $\oplus(s : \mathsf{N}' \odot (\Sigma \cup \{\sigma\}))$ (Line 23).

The loops in our algorithm guarantee that an exhaustive (linear) search through a normative state takes place, checking if the new normative position is in conflict with any existing prohibitions/obligations, possibly updating the annotations of any conflicting prohibitions. The algorithm is correct in that, for a given normative position and a normative state, it provides a (possibly empty) set of updates adding new normative positions to Δ or changing existing normative positions, ensuring that Δ remains conflict-free, that is, after the commands in Cs are applied to a (possibly empty) conflict-free Δ, Δ remains conflict-free. The prohibitions in Δ, however, all have annotations recording how they unify with existing obligations. The annotations can be empty, though: this is the case when we have a ground prohibition or a prohibition which does not unify/conflict with any obligation. Permissions do not affect our algorithm and they are appropriately dealt with (Line 3). Any attempt to insert a ground prohibition which conflicts, yields an empty set of updates in normative positions (Line 4). When a new obligation is being added then the algorithm guarantees that all prohibitions are considered (Lines 15–24), leading to updates for the removal of conflicting ground prohibitions or the change of annotations of non-ground prohibitions. The algorithm always terminates: the loops are over a finite set Δ and the *conflict* checks always terminate. The complexity of the algorithm is linear: the set Δ is only examined once for each possible case of normative position to be added.

We illustrate how our algorithm works with an example from the scenario of Section 2. Let us suppose that agents are participating in a negotiation activity that we will call d. For reasons of space, let us assume that agent a_1 informing agent a_2 of $p(I,P)$ counts as a_1 paying the amount P for item I to a_2. Furthermore, let us suppose that agent a_1 has not fulfilled a previous contract with a_2 while participating in this new negotiation activity yielding new pending obligations. We may forbid agent a_1 to obtain more obligations with regard to agent a_2 until it fulfills the currently pending ones.[8] Then let us suppose we invoke the algorithm with the following parameters:

1. The update $\oplus(s : \mathsf{N})$ to be processed is

$$\oplus(d : prh(inform(a_1, r_1, a_2, r_2, p(X,Y), Z)))$$

The update represents a prohibition on agent a_1 to pay agent a_2 for *any* item at *any* price.

2. The current normative state Δ of activity d is

$$\left\{ \begin{array}{l} \langle obl(inform(a_1, r_1, a_2, r_2, p(1,2), 3))), 3\rangle, \\ \langle obl(inform(a_1, r_1, a_2, r_2, p(10,20), 30))), 30\rangle \end{array} \right\}$$

In our example, agent a_1 is obliged to pay agent a_2 2 Euros for item 1 and 20 Euros for item 10.

The algorithm will provide us with a set Cs of updates required to accommodate the input update, ensuring Δ is kept conflict-free.

[8] It may be the case that further correcting actions should be performed.

Let us assume that the current time is 40, thus both normative positions in Δ are active. The example is handled by the third case (Lines 8–14): the set Σ is initialized to $\emptyset$ (Line 10) then a loop is executed (Line 11) where each of the normative positions in Δ is considered in turn, being checked for a conflict with the input update. In our example, we have:

$$\mathbf{conflict}(prh(inform(a_1, r_1, a_2, r_2, p(X,Y), Z)),$$
$$obl(inform(a_1, r_1, a_2, r_2, p(1,2), 3)), \sigma_1)$$
$$\mathbf{conflict}(prh(inform(a_1, r_1, a_2, r_2, p(X,Y), Z)),$$
$$obl(inform(a_1, r_1, a_2, r_2, p(10,20), 30)), \sigma_2)$$

where $\sigma_1 = \{X/1, Y/2, Z/3\}$ and $\sigma_2 = \{X/10, Y/20, Z/30\}$, hence the algorithm computes $\Sigma = \{\sigma_1, \sigma_2\}$. In Line 13, the set of updates Cs is computed, that is,

$$\mathsf{Cs} = \{\oplus(d : prh(inform(a_1, r_1, a_2, r_2, p(X,Y), Z)) \odot \{\sigma_1, \sigma_2\})\}$$

The annotation ensures that the prohibition does not have values which overlap with those of the obligations.

Although we propose to curtail prohibitions, the same machinery introduced above can be used to define the curtailment of obligations or permissions instead. Different policies (that is, which normative position should be preserved and which one should be curtailed) are dependent on the intended deontic semantics and requirements of the systems addressed.

5. A distributed architecture for normative structures

In this section we propose an architecture, expanded and adapted from (García-Camino, Rodrıguez-Aguilar, & Vasconcelos, 2008), to address the regulation of the behavior of autonomous agents and the management of the normative state(s) of the MAS, including the management of normative positions and the resolution of normative conflicts. We assume the existence of heterogeneous, selfish agents that interact in order to pursue their goals – we do not have control on these agents' internal functioning, nor can we anticipate it.

We propose to extend AMELI (Esteva, Rosell, Rodríguez-Aguilar, & Arcos, 2004), an agent-based infrastructure that allows the enactment of open multi-agent systems as electronic institutions. Such extension allows us to inherit all the features of AMELI, including, very importantly, its capability of regulating open multi-agent systems. The architecture of AMELI is divided into three (logical) layers, illustrated in Fig. 3, namely:

- Autonomous agent layer – the set of external agents taking part in the MAS.
- Social layer – an infrastructure that mediates and facilitates agents' interactions while enforcing institutional rules.
- Communication layer – it provides a reliable and orderly transport service.

External agents intending to communicate with other external agents need to redirect their messages through the social layer, which is in charge of forwarding the messages (attempts of communication) to the communication layer. In specified conditions, erroneous or illicit messages (those that do not comply with the rules of the institution) may be rejected by the social layer in order to prevent them from arriving at their addressees.

The social layer presented in (Esteva et al., 2004) is a multi-agent system itself and the agents belonging to it are called internal agents. We propose to extend AMELI: (i) by adding a new type of agent to the social layer, the so-called *normative managers* (represented as circles labelled $NM_1, \ldots, NM_p$ in Fig. 3); and (ii) by adding protocols to coordinate this new type of agent with the rest of internal agents to manage normative transitions and resolve

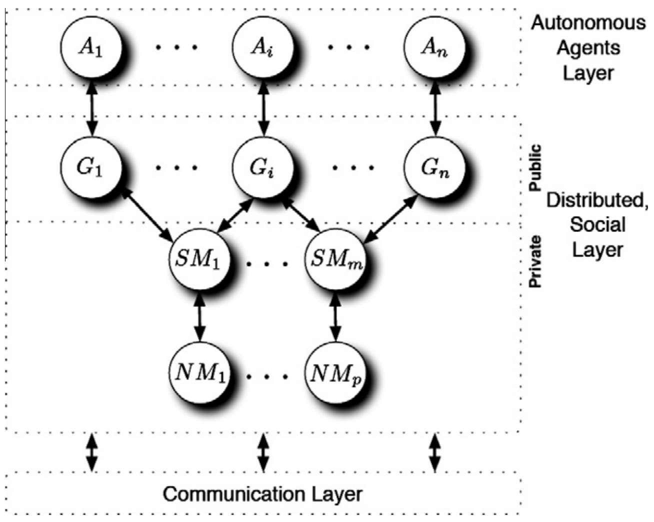

Fig. 3. AMELI$^+$ architecture.

conflicts. We call AMELI$^+$ the resulting architecture. Thus, in AMELI$^+$, internal (administrative) agents are of the following types:

- Governor (*G*) – This is an internal agent representing an external agent. It maintains the external agent's social state and informs the external agent about its normative positions. Very importantly, the governor agent acts as an intermediary for message exchanges among external agents, deciding on whether an external agent's attempt at sending a message to another external agent is norm-compliant and, if so, forwarding this message to the appropriate external agent (who will receive the message via its governor agent). We require one governor agent per external agent.
- Scene Manager (*SM*) – This is an internal agent maintaining the state of the activity[9], notifying any changes to normative managers and resolving conflicts.
- Normative Manager (*NM*) – This new type of internal agent receives normative commands and may fire one or more normative transitions.

In principle, only one *NM* is needed to manage all the normative transitions. However, in order to build large MAS and avoid bottlenecks, we propose the distribution of normative transitions into several *NM*s. The choice of the granularity of the normative layer, *i.e.* to choose from one single *NM* to one *NM* per normative transition, is an important design decision left to MAS designers.

AMELI$^+$ computationally realizes normative structures, as described in Section 2, making use of the conflict resolution algorithm in Section 4. On the one hand, *NM*s handle normative transitions by coordinating their activities with *SM*s, who manage their activities' normative states. Therefore, the links connecting normative transitions to normative states map to coordinations between *NM*s and *SM*s. On the other hand, *SM*s are responsible for solving conflicts by running the conflict resolution algorithm in Section 4. The architecture scales up well, as the administrative agents can run in different machines, so we have, in principle, as many agents as we have hardware to run them on (and more hardware can be added at will). Moreover, an agent-based architecture provides additional benefits as the agents can be endowed with arbitrary functionalities, *e.g.*, our normative managers are able to run the conflict detection checks and the conflict resolution algorithm, and our governor agents can be endowed with sophisticated forms of dialogue with external agents, the former warning the latter about the consequence of their non-norm-compliant behavior (as explored in (García-Camino et al., 2006)).

6. Related work

Within agent-mediated electronic commerce (He, Jennings, & Leung, 2003; Sierra, 2004), various challenges arise when building open (that is, components come and go during the system's execution), heterogeneous (that is, individual components have been implemented by different parties, using disparate architectures and assorted programming languages and technologies), and distributed (that is, the components run in various machines, geographically distant, communicating via message-passing) systems that will carry out sophisticated many-party interactions to provide goods and/or services.

Norms offer a useful and powerful abstraction with which to specify and regulate such systems, excluding disruptive or antisocial behavior without prescribing the design of individual agents or restricting their autonomy (Dignum, 1999; Vasconcelos et al., 2009). Norms provide a generic account of individual behaviors; if all agents have norm-regulated behaviors, then we can offer guarantees about the system as a whole. Similarly, norms provide a declarative statement of how agents are expected to behave, and so offer agents a way to predict the behavior of others in response to requests, the provision of information, etc. There is a wealth of socio-philosophical and logic-theoretical literature on the subject of norms (*e.g.*, Sergot, 2001; Shoham & Tennenholtz, 1995; von Wright, 1963), and, more recently, much attention is being paid to more pragmatic and implementation-oriented aspects of norms, that is, how norms can be given a computational interpretation and how norms can be taken into account in the design and execution of MASs (*e.g.*, Artikis, Kamara, Pitt, & Sergot, 2005; Cranefield, 2005; Viganò, Fornara, & Colombetti, 2006; García-Camino et al., 2006).

However, to the best of our knowledge, very few authors have provided implementations managing and reasoning about normative positions in a distributed manner. In (Esteva et al., 2004; Minsky, 2005), two languages are presented for the distributed enforcement of normative positions in MASs. In (Esteva et al., 2004) an obligation is a first-order formula whose intuitive meaning is that if grounded utterances matching the given utterance schema are uttered in an activity, and given conditional expressions are satisfied, then, further grounded utterances matching the given utterance schema satisfying the given conditional expressions must be uttered in the corresponding activity.

In (Minsky, 2005; Minsky & Ungureanu, 2000) the authors propose law-governed interaction (LGI), a decentralized coordination and control mechanism for distributed systems. This middleware allows a possible large, heterogeneous and open set of actors to interact governed by a given policy, called the *interaction law*. In order to enforce the interaction law, a component called *controller* is associated to each actor. The controller is entrusted to mediate the interaction of its actor with others, it decides, by interpreting the active law, how to react to messages sent and received by its actor.

However, in both approaches (Esteva et al., 2004; Minsky, 2005), each agent has a local message interface that forwards legal messages according to a set of norms. Since these interfaces are local to each agent, norms can only be expressed in terms of actions of that agent. This is a serious disadvantage, *e.g.*, when one needs to activate an obligation to one agent due to a certain message of another one.

The work presented in (Hübner, Boissier, Kitio, & Ricci, 2009) proposes Organizational Artifacts for multi-agent systems (ORA4-MAS). This approach may be seen as a distributed generalization of tuple-spaces but using objects and providing further functionalities. Its authors argue that instead of providing a "dialogical environment" as in the architecture presented in this paper and

[9] Activities are also referred to as scenes following the terminology of AMELI.

predecessors, *e.g.*, (Esteva et al., 2004), they provide a "work environment". However, as shown in (García-Camino et al., 2006), valid dialogical actions may trigger the performance of further non-dialogical actions in the coordination level. More languages for the coordination level are proposed in (García-Camino, 2009).

This paper focuses on the normative level, which connects several activities taking place in the coordination level. Instead of proposing that activities (or workspaces in the terminology of Hübner et al., 2009) exchange information directly, we decouple as much as we can propagation of normative positions from the norm enforcement taking place in the activities and, as mentioned above, our approach also includes the translation from dialogical actions into non-dialogical ones. We admit that the languages proposed in (García-Camino, 2009) for the coordination level do not use an object-oriented approach but straightforward modifications would allow expressing non-dialogical actions that would interface the coordination level with object-oriented languages.

(Sartor, 1992) treats normative conflicts from the point of view of legal theory and suggests a way to order the norms involved. This idea is implemented in (García-Camino et al., 2007) but requires a central resource for normative position maintenance. A mechanism for conflict resolution of normative positions that is somewhat similar to ours has been sketched in (Kollingbaum & Norman, 2004). However, their mechanism is presented only informally and uses instantiation graphs – their high computational complexity only allows simple scenarios to be addressed. Additionally, that work only considers individual norm-compliant agents, and not an institutional (or organizational) infrastructure for open MASs. Our approach to detecting normative conflicts can capture the three forms of conflict/inconsistency of (Ross, 1958), *viz.* total-total, total-partial and intersection, respectively, when the permission entails the prohibition, when the prohibition entails the permission and when they simply overlap. The normative position management algorithm we presented in Section 4 is an adaptation and extension of the work presented in (Vasconcelos et al., 2007, 2009), also providing an investigation into deontic modalities for representing normative concepts (Dignum, 1999; Sergot, 2001).

7. Conclusions, discussion & future work

In this paper, we have introduced *normative structures* for the management of normative positions. Ours is a useful approach because it allows the separation of normative and procedural concerns. We have also proposed an algorithm for *run-time conflict resolution*. The normative structures presented are computationally realized via a distributed architecture, AMELI$^+$, described in this paper. Our original requirements, namely, to regulate the operation of an open multi-agent system with multiple, concurrent, distributed and inter-related activities, are met by our distributed normative model.

Although we formerly introduced the notion of normative structure in (Gaertner, García-Camino, Noriega, Rodriguez-Aguilar, & Vasconcelos, 2007), this paper extends early results along several directions. Firstly, the semantics of normative positions have been clarified. With this aim, we have resorted to formally defining the notion of norm violation. Additionally, normative structures handle the normative states of activities at the coordination level. Finally, the new algorithm for conflict resolution in Section 4 differs in several ways from its previous version, namely, (i) it addresses normative positions which are relative to activities/scenes; (ii) it is more compact; and (iii) instead of updating the input set of normative positions Δ, it uses Δ to create *commands* (or updates to Δ) which when applied to Δ will ensure conflict freedom. The third point is essential as the commands can be performed in a distributed fashion.

The contributions we have presented open many possibilities for future work. We are currently working on a generalization of normative structures to make them operational with different coordination models, with richer deontic content and on top of different computational realizations of regulated MASs. As a first step in this direction we are taking advantage of the decoupling between interaction protocols and declarative normative guidance that the normative structure makes available, to provide a normative layer for electronic institutions (as defined in Arcos et al., 2005). We expect such combination will endow electronic institutions with a more flexible and expressive normative environment.

Furthermore, we want to extend our model along several directions. We would like to handle negation and arbitrary constraints (similarly to García-Camino, 2009) as part of the norm language, and in particular temporal aspects such as deadlines for normative positions and their extensions. We also want to accommodate multiple, hierarchical norm authorities based on roles, along the lines of (Cholvy & Cuppens, 1995) and power relationships as suggested in (Carabelea, Boissier, & Castelfranchi, 2004). Finally, we want to capture in the conflict resolution algorithm different semantics relating the deontic notions by supporting different axiomatizations (*e.g.*, relative strength of prohibition versus obligation, default deontic notions, deontic inconsistencies).

Acknowledgements

This work was partially supported by research projects IEA (TIN2006-15662-C02-01), AT (CONSOLIDER CSD2007-0022) and EVE (TIN2009-14702-C02-01). J. A. Rodríguez-Aguilar acknowledges the support of the José Castillejo Programme (JC2008-00337). This research continued through W. W. Vasconcelos's participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence (http://www.usukita.org). The research was also funded by the Secretaría de Estado de Investigación, the EU FEDER funds, and the Generalitat de Catalunya (grant 2009-SGR-1434).

References

Aldewereld, H., Vázquez-Salceda, J., Dignum, F., & Meyer, J.-J. C. (2005). Norm compliance of protocols in electronic institutions. In *Proceedings of the fourth international joint conference on autonomous agents & multiagent systems* (pp. 1291–1292). New York, NY, USA: ACM.
Apt, K. R. (1997). *From logic programming to prolog*. U.K.: Prentice-Hall.
Arcos, J. L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J. A., & Sierra, C. (2005). Engineering open environments with electronic institutions. *Journal on engineering applications of artificial intelligence, 18*(2), 191–204.
Artikis, A., Kamara, L., Pitt, J., & Sergot, M. (2005). A protocol for resource sharing in norm-governed ad-hoc networks. In *Declarative agent languages and technologies II. LNCS* (Vol. 3476, pp. 221–238). Springer-Verlag.
Boella, G., Pigozzi, G., & Van der Torre, L. (2009). Five guidelines for normative multiagent systems. In G. Governatore (Ed.), *Legal Knowledge and Information Systems. JURIX 2009* (pp. 21–30) October 22–24, 2009, Amsterdam: IIOS Press. ISSN 0105-8517.
Carabelea, C., Boissier, O., & Castelfranchi, C. (2004). Using social power to enable agents to reason about being part of a group. In *Fifth international workshop on engineering societies in the agents world (ESAW)* (pp. 166–177).
Cholvy, L., & Cuppens, F. (1995). Solving normative conflicts by merging roles. In *Fifth international conference on artificial intelligence and law* (pp. 201–209). Washington, USA .
Cranefield, S. (2005). A rule language for modelling and monitoring social expectations in multi-agent systems. Technical Report 2005/01, University of Otago, February 2005.
Cristian, F. (1991). Understanding fault-tolerant distributed systems. *Communications of the ACM, 34*(2), 56–78.
Dignum, F. (1999). Autonomous agents with norms. *Artificial intelligence and law, 7*(1), 69–79.
Elhag, A. A. O., Breuker, J. A. P. J., & Brouwer, B. W. (2000). On the formal analysis of normative conflicts. *Information & Communications Technology Law, 9*(3), 207–217.
Esteva, M., Rosell, B., Rodríguez-Aguilar, J. A., & Arcos, J. L. (2004). AMELI: An agent-based middleware for electronic institutions. In *AAMAS 2004. Third international joint conference on autonomous agents and multiagent systems* (pp. 236–243). ACM.

Esteva, M., Vasconcelos, W. W., Sierra, C., & Rodríguez-Aguilar, J. A. (2004). Norm consistency in electronic institutions. In *XVII Brazilian symposium on artificial intelligence (SBIA'04). LNAI* (Vol. 3171, pp. 494–505). Springer-Verlag.

Fitting, M. (1990). *First-order logic and automated theorem proving*. New York, U.S.A.: Springer-Verlag.

Gaertner, D., García-Camino, A., Noriega, P., Rodriguez-Aguilar, J.-A., & Vasconcelos, W. W. (2007). Distributed norm management in regulated multiagent systems. In *AAMAS '07: Proceedings of the sixth international joint conference on Autonomous agents and multiagent systems* (pp. 1–8). New York, NY, USA: ACM.

García-Camino, A. (2009). Normative Regulation of Open Multi-Agent Systems. PhD thesis, Universitat Autònoma de Barcelona, 2009. Number 35 in IIIA Monograph Series.

García-Camino, A., Noriega, P., & Rodríguez-Aguilar, J.A. (2007). An algorithm for conflict resolution in regulated compound activities, In G. M. P. O'Hare, A. Ricci, M. J. O'Grady, & O. Dikenelli (Eds.), Engineering societies in the agents world VII, seventh international workshop, ESAW 2006. Revised Selected and Invited Papers, vol. 4457 (pp. 193–208), Dublin, Ireland, September 6–8, Springer.

García-Camino, A., Rodríguez-Aguilar, J. A., Sierra, C., & Vasconcelos, W. W. (2009). Constraint rule-based programming of norms for electronic institutions. *Autonomous agents and multi-agent systems, 18*(1), 186–217.

García-Camino, A., Rodríguez-Aguilar, J. A., & Vasconcelos, W. W. (2008). A distributed architecture for norm management in multi-agent systems. In *Proceedings of the AAMAS workshop on coordination, organization, institutions and norms in agent systems (COIN). Lecture notes in computer science* (Vol. 4870, pp. 275–286). Springer-Verlag.

García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., & Vasconcelos, W. W. (2006). A distributed architecture for norm-aware agent societies. In *Declarative agent languages and technologies III. LNAI* (Vol. 3904, pp. 89–105). Springer.

Giunchiglia, F., & Serafini, L. (1994). Multi-language hierarchical logics or: How we can do without modal logics. *Artificial intelligence, 65*(1), 29–70.

He, M., Jennings, N. R., & Leung, H.-F. (2003). On agent-mediated electronic commerce. *IEEE transactions on knowledge and data engineering, 15*(4), 985–1003.

Hübner, J. F., Boissier, O., Kitio, R., & Ricci, A. (2009). Instrumenting multi-agent organisations with organisational artifacts and agents: "giving the organisational power back to agents". *Autonomous agents and multi-agent systems, 20*(3).

Kollingbaum, M., & Norman, T. (2004). Strategies for resolving norm conflict in practical reasoning. In *Proceedings of the ECAI workshop on coordination in emergent agent societies* (pp. 25–36).

Minsky, N. H. (2005). Law Governed Interaction (LGI): A distributed coordination and control mechanism (an introduction, and a reference manual). Technical report, Rutgers University.

Minsky, N. H., & Ungureanu, V. (2000). Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM transactions on software engineering and methodology, 9*(3), 273–305.

Ross, A. (1958). *On law and justice*. Stevens & Sons.

Sartor, G. (1992). Normative conflicts in legal reasoning. *Artificial intelligence and law, 1*(2-3), 209–235.

Searle, J. (1969). *Speech acts*. London: Cambridge University Press.

Searle, J. (1995). *The construction of social reality*. London: The Penguin Press.

Sergot, M. (2001). A computational theory of normative positions. *ACM transactions of computational logic, 2*(4), 581–622.

Shoham, Yoav, & Tennenholtz, Moshe (1995). On social laws for artificial agent societies: Off-line design. *Artificial intelligence, 73*(1–2), 231–252.

Sierra, Carles (2004). Agent-mediated electronic commerce. *Autonomous agents and multi-agent systems, 9*(3), 285–301.

Vasconcelos, W.W., Kollingbaum, M.J., & Norman, T.J. (2007). Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proceedings of sixth international joint conference on autonomous agents & multiagent systems (AAMAS 2007)* (pp. 632–639), Hawai'i, U.S.A., May 2007. IFAAMAS.

Vasconcelos, W. W., Kollingbaum, M. J., & Norman, T. J. (2009). Normative conflict resolution in multi-agent systems. *Autonomous agents and multi-agent systems, 19*(2), 124–152.

Viganò, F., Fornara, N., & Colombetti, M. (2006). An event driven approach to norms in artificial institutions. In *Coordination, organizations, institutions, and norms in multi-agent systems (COIN). LNCS* (Vol. 3913, pp. 142–154). Springer Verlag.

von Wright, G. H. (1963). *Norm and action: A logical inquiry*. London: Routledge and Kegan Paul.

Wooldridge, M. (2002). *An introduction to multiagent systems*. Chichester, UK: John Wiley & Sons.