Resolving Conflict and Inconsistency in Norm-Regulated Virtual Organizations

Wamberto Vasconcelos
Dept. of Computing Science
University of Aberdeen
Aberdeen AB24 3UE
United Kingdom
wvasconcelos@acm.org

Martin J. Kollingbaum
Dept. of Computing Science
University of Aberdeen
Aberdeen AB24 3UE
United Kingdom
mkolling@csd.abdn.ac.uk

Timothy J. Norman
Dept. of Computing Science
University of Aberdeen
Aberdeen AB24 3UE
United Kingdom
tnorman@csd.abdn.ac.uk

ABSTRACT

Norm-governed virtual organizations define, govern and facilitate coordinated resource sharing and problem solving in societies of agents. With an explicit account of norms, *openness* in virtual organizations can be achieved: new components, designed by various parties, can be seamlessly accommodated. We focus on virtual organizations realised as multi-agent systems, in which human and software agents interact to achieve individual and global goals. However, any realistic account of norms should address their dynamic nature: norms will change as agents interact with each other and their environment. Due to the changing nature of norms or due to norms stemming from different virtual organizations, there will be situations when an action is simultaneously permitted and prohibited, that is, a *conflict* arises. Likewise, there will be situations when an action is both obliged and prohibited, that is, an *inconsistency* arises. We introduce an approach, based on first-order unification, to detect and resolve such conflicts and inconsistencies. In our proposed solution, we annotate a norm with the set of values their variables should *not* have in order to avoid a conflict or an inconsistency with another norm. Our approach neatly accommodates the domain-dependent interrelations among actions and the indirect conflicts/inconsistencies these may cause. More generally, we can capture a useful notion of inter-agent (and inter-role) *delegation* of actions and norms associated to them, and use it to address conflicts/inconsistencies caused by action delegation. We illustrate our approach with an e-Science example in which agents support Grid services.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Applications and Expert Systems—*Law*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

General Terms

Algorithms, Theory

Keywords

Artificial social systems: conventions, norms, institutions.

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

1. INTRODUCTION

Virtual organizations (VOs) facilitate coordinated resource sharing and problem solving involving various parties geographically remote [9]. VOs define and regulate interactions (thus facilitating coordination) among software and/or human agents that communicate to achieve individual and global goals [16]. VOs are realised as multi-agent systems and a most desirable feature of such systems is *openness* whereby new components designed by other parties are seamlessly accommodated. The use of norms, that is, prohibitions, permissions and obligations, in the specification and operation of multi-agent systems (MASs) is a promising approach to achieving openness [2, 4, 5, 6]. Norms regulate the observable behaviour of self-interested, heterogeneous software agents, designed by various parties who may not entirely trust each other [3, 24]. However, norm-regulated VOs may experience problems when norms assigned to their agents are in conflict (*i.e.*, an action is simultaneously prohibited and permitted) or inconsistent (*i.e.*, an action is simultaneously prohibited and obliged).

We propose a means to automatically detect and solve conflict and inconsistency in norm-regulated VOs. We make use of first-order term unification [8] to find out if and how norms overlap in their influence (*i.e.*, the agents and values of parameters in agents' actions that norms may affect). This allows for a fine-grained solution whereby the influence of conflicting or inconsistent norms is *curtailed* for particular sets of values. For instance, norms "agent x is permitted to *send_bid*$(ag_1, 20)$" and "agent ag_2 is prohibited from doing *send_bid*(y, z)" (where x, y, z are variables and ag_1, $ag_2, 20$ are constants) are in conflict because their agents, actions and terms (within the actions) unify. We solve the conflict by annotating norms with sets of values their variables *cannot* have, thus curtailing their influence. In our example, the conflict is avoided if we require that variable y cannot be ag_1 and that z cannot be 20.

This paper is organized as follows. In the next section we provide a minimalistic definition for norm-regulated VOs. In section 3 we formally define norm conflicts, and explain how they are detected and resolved. In section 4 we describe how the machinery of the previous section can be adapted to detect and resolve norm inconsistencies. In section 5 we describe how our curtailed norms are used in norm-aware agent societies. In section 6 we explain how our machinery can be used to detect and solve *indirect* conflicts/inconsistencies, that is, those caused via relationships among actions; we extend and adapt the machinery to accommodate the *delegation* of norms. In section 7 we illustrate our approach with an example of norm-regulated software agents serving the Grid. In section 8 we survey related work and in section 9 we discuss our contributions and give directions for future work.

978-81-904262-7-5 (RPS)

2. VIRTUAL ORGANIZATIONS

Virtual organizations [17] allow various parties to come together to share resources and engage in problem solving. This paradigm has found strong applications in Web-service orchestration [14], e-Science [16] and the Grid [9]. VOs, in their most generic formulation, can be seen as coordination artifacts, allowing software and human agents to engage in sophisticated forms of interaction.

We formally represent our VOs as finite-state machines in which the actions of individual agents label the edges between discrete states. This provides us with a "lowest common denominator": there are much more sophisticated, convenient and expressive ways to represent interactions among agents (*e.g.*, AUML [19] and electronic institutions [20], to name a few), but for the sake of generalising our approach, we shall assume any higher-level formalism can be mapped onto a finite-state machine (possibly with some loss of expressiveness). We show in Figure 1 a simple VO graphically represented as a finite-state machine[1]. The labels on the edges connecting the states are first-order atomic formulae, denoted generically as φ; they stand for actions performed by individual agents. We define our virtual organizations as follows:

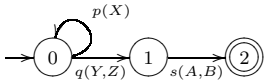

Figure 1: Sample VO as a Finite-State Machine

DEF. 1. *A virtual organization $\mathcal{I}$ is the triple $\langle S, s_0, E, T\rangle$ where $S = \{s_1, \ldots, s_n\}$ is a finite and non-empty set of states, $s_0 \in S$ is the initial state, E is a finite set of edges $(s, s', \varphi), s, s' \in S$ connecting s to s' with a first-order atomic formula φ as a label, and $T \subseteq S$ is the set of terminal states.*

Notice that edges are directed, so $(s, t, \varphi) \neq (t, s, \varphi)$. The sample VO of Figure 1 is formally represented as $\mathcal{I} = \langle\{0, 1, 2\}, 0, \{(0, 0, p(X)), (0, 1, q(Y, Z)), (1, 2, s(A, B)\}, \{2\}\rangle$. We assume an implicit existential quantification on any variables in φ, so that, for instance, $s(A, B)$ stands for $\exists A, B\ s(A, B)$.

VOs should allow for two kinds of non-determinism corresponding to choices autonomous agents can make, *viz.*, *i)* the one arising when there is more than one edge leaving a state; and *ii)* the one arising from variables in the formulae φ labelling an edge, for which the agent carrying out the action instantiates. These kinds of non-determinism are desirable as they help define generic and flexible coordination mechanisms.

Another important concept we use is the *roles* of agents in VOs. Roles, as exploited in, for instance, [18] and [20], help us abstract from individual agents and define a pattern of behaviour to which any agent that adopts a role ought to conform. Moreover, all agents with the same role are guaranteed the same rights, duties and opportunities. We shall make use of two finite, non-empty sets, $Agents = \{ag_1, \ldots, ag_n\}$ and $Roles = \{r_1, \ldots, r_m\}$, representing, respectively, the sets of agent identifiers and role labels. We refer generically to first-order terms, *i.e.*, constants, variables, and (nested) functions as τ.

2.1 Semantics of VOs

The specification of a VO as a finite-state machine gives rise to a possibly infinite set of histories of computational behaviours, in which the actions labelling the paths from the initial state to a final state are recorded. Although the actions comprising a VO are carried out distributedly, we propose an explicit global account of all events. In practice, this can be achieved if we require individual agents to declare/inform whatever actions they have carried out; this assumes trustworthy agents, naturally[2].

In order to record the authorship of the action, we annotate the formulae with the agents' unique identification. Our explicit global account of all events is a set of ground atomic formulae φ, that is, we only allow constants to appear as terms of formulae. Each formula is a truthful record of an action specified in the VO. Notice, however, that in the VO specification we do not restrict the syntax of the formulae: variables may appear in them, and when an agent performs an actual action then any variables of the specified action must be assigned values. We thus define:

DEF. 2. *A global execution state of a VO, denoted as Ξ, is a finite, possibly empty, set of tuples $\langle a : r, \bar{\varphi}, t\rangle$ where $a \in Agents$ is an agent identifier, $r \in Roles$ is a role label, $\bar{\varphi}$ is a ground first-order atomic formula, and $t \in \mathbb{N}$ is a time stamp.*

For instance, $\langle ag_1$: $buyer, p(a, 34), 20\rangle$ states that agent ag_1 adopting role $buyer$ performed action $p(a, 34)$ at instant 20. Given a VO $\mathcal{I} = \langle S, s_0, E, T\rangle$, an execution state Ξ and a state $s \in S$, we can define a function which obtains a possible next execution state, *viz.*, $h(\mathcal{I}, \Xi, s) = \Xi \cup \{\langle a$:$r, \bar{\varphi}, t\rangle\}$, for one $(s, s', \varphi) \in E$. Such function h must address the two kinds of non-determinism above, as well as the choice on the potential agents that can carry out the action and their adopted roles. We also define a function to compute the set of all possible execution states, $h^*(\mathcal{I}, \Xi, s) = \{\Xi \cup \{\langle a$: $r, \bar{\varphi}, t\rangle\}|(s, s', \varphi) \in E\}$.

2.2 Norm-Regulated VOs

We advocate a *separation of concerns* whereby the virtual organization is complemented with an explicit and separate set of norms that further regulates the behaviour of agents as they take part in the enactment of an organization. The freedom of choice given to agents (captured via the non-determinism of VOs, explained above) must be curtailed in some circumstances. For instance, we might need to describe that whoever carried out φ is obliged to carry out φ', so that if there is a choice point in which φ' appears as a label of an edge, then that edge should be followed.

Rather than embedding such normative aspects into the agents' design (say, by explicitly encoding normative aspects in the agents' behaviour) or into the VO itself (say, by addressing exceptions and deviant behaviour in the mechanism itself), we adopt the view that a VO should be supplemented with a separate set of norms that further regulates the behaviour of agents as they take part in the enactment of the organization. This separation of concerns should facilitate the design of MASs; however, the different components (VOs and norms) must come together at some point in the design process. Our norms are defined as below:

DEF. 3. *A norm, generically referred to as ν, is any construct of the form $\mathsf{O}_{\tau:\tau'}\varphi$, $\mathsf{P}_{\tau:\tau'}\varphi$, or $\mathsf{F}_{\tau:\tau'}\varphi$, where τ, τ' are either variables or constants and φ is a first-order atomic formula.*

We adopt the notation of [18]: $\mathsf{O}_{\tau:\tau'}\varphi$ represents an obligation on agent τ taking up role τ' to bring about φ; we recall that τ, τ' are variables, constants and functions applied to (nested) terms. $\mathsf{P}_{\tau:\tau'}\varphi$ and $\mathsf{F}_{\tau:\tau'}\varphi$ stand for, respectively, a permission and a prohibition on agent τ, playing role τ' to bring about φ. We shall assume that sorts are used to properly manipulate variables for agent identifiers and role labels.

We propose to formally represent the normative positions of all agents enacting a VO. By "normative position" we mean the "social burden" associated to individuals [12], that is, their obligations, permissions and prohibitions:

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

[2] Non-trustworthy agents can be accommodated in this proposal, if we associate to each of them a *governor agent* which supervises the actions of the external agent and reports on them. This approach was introduced in [12] and is explained in section 5.

DEF. 4. *A global normative state Ω is a finite and possibly empty set of tuples $\omega = \langle \nu, t_d, t_a, t_e \rangle$ where ν is a norm as above and $t_d, t_a, t_e \in I\!N$ are, respectively, the time when ν was declared (introduced), when ν becomes active and when ν expires, $t_d \leq t_a < t_e$.*

It is worth noticing that we do not require the atomic formulae of norms to be ground: there could be variables in them. We assume an implicit universal quantification on the variables A, R of norms $\mathsf{X}_{A:R}\varphi$ (for the deontic modalities $\mathsf{X} \in \{\mathsf{O}, \mathsf{P}, \mathsf{F}\}$), so that, for instance, $\mathsf{P}_{A:R}p(X, b, c)$ stands for $\forall A \in Agents.\forall R \in Roles.\exists X.\mathsf{P}_{A:R}p(X, b, c)$. We also refer to the tuples in Ω as norms.

Global normative states complement the execution states of VOs with information on the normative positions of individual agents. We can relate them via a function to obtain a norm-regulated next execution state of a VOs, that is, $g(\mathcal{I}, \Xi, s, \Omega, t) = \Xi'$, t standing for the time of the update. For instance, we might want all prohibited actions to be excluded from the next execution state, that is, $g(\mathcal{I}, \Xi, s, \Omega, t) = \Xi \cup \{\langle a : r, \bar{\varphi}, t \rangle\}, (s, s', \varphi) \in E$ and $\langle \mathsf{F}_{a:r}\varphi, t_d, t_a, t_e \rangle \notin \Omega, t_a \leq t \leq t_e$. We might equally wish that only permitted actions be chosen for the next execution state. We do not legislate, or indeed recommend, any particular way to regulate VOs. We do, however, offer simple underpinnings to allow arbitrary policies to be put in place.

In the same way that a normative state is useful to obtain the next execution state of a VO, we can use an execution state to update a normative state. For instance, we might want to remove any obligation specific to an agent and role, which has been carried out by that specific agent and role, that is, $f(\Xi, \Omega) = \Omega - Obls$, $Obls = \{\langle \mathsf{O}_{a:r}\varphi, t_d, t_a, t_e \rangle \in \Omega | \langle a : r, \bar{\varphi}, t \rangle \in \Xi\}$.

The management (*i.e.*, creation and updating) of global normative states is an interesting area of research. A simple but useful approach is reported in [11]: production rules generically depict how norms should be updated to reflect what agents have done and which norms currently hold. In this paper our focus is not to propose how Ω's should be managed; we assume some mechanism which does that.

3. NORM CONFLICTS

We now define means to detect and resolve norm conflicts and inconsistencies. We make use of the concept of *unification* [1, 8] of first-order terms τ, *i.e.*, constants, variables or (nested) functions with terms as parameters. Initially we define substitutions:

DEF. 5. *A substitution σ is a finite and possibly empty set of pairs x/τ, where x is a variable and τ is a term.*

We define the application of a substitution as:

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. $p^n(\tau_0, \ldots, \tau_n) \cdot \sigma = p^n(\tau_0 \cdot \sigma, \ldots, \tau_n \cdot \sigma)$.
4. $(\mathsf{X}_{\tau_1:\tau_2}\varphi) \cdot \sigma = \mathsf{X}_{(\tau_1 \cdot \sigma):(\tau_2 \cdot \sigma)}(\varphi \cdot \sigma)$
5. $\langle \nu, t_d, t_a, t_e \rangle \cdot \sigma = \langle (\nu \cdot \sigma), t_d, t_a, t_e \rangle$

Where X generically refers to any of the deontic modalities $\mathsf{O}, \mathsf{P}, \mathsf{F}$. Unification between two terms τ, τ' consists of finding a substitution σ (also called, in this context, the *unifier* of τ and τ') such that $\tau \cdot \sigma = \tau' \cdot \sigma$. Many algorithms have been proposed to solve the unification problem, a fundamental issue in automated theorem proving [8], and more recent work provides very efficient ways to obtain unifiers. We shall make use of the following definition:

DEF. 6. *Relationship $unify(\tau, \tau', \sigma)$ holds iff there is a possibly empty σ such that $\tau \cdot \sigma = \tau' \cdot \sigma$.*

We also define the unification of atomic formulae as $unify(p^n(\tau_0, \ldots, \tau_n), p^n(\tau'_0, \ldots, \tau'_n), \sigma)$ which holds iff $\tau_i \cdot \sigma = \tau'_i \cdot \sigma, 0 \leq i \leq n$. The *unify* relationship checks if a substitution σ is indeed a unifier for τ, τ' but it can also be used to find such σ. We assume that *unify* is a suitable implementation of a unification algorithm which *i)* always terminates (possibly failing, if a unifier cannot be found); *ii)* is correct; and *iii)* has a linear computational complexity.

3.1 Conflict Detection

A norm conflict arises when an atomic formula labelling an edge in the VO, *i.e.* an action, is simultaneously permitted and prohibited [13]. In this case, both norms are in conflict with regard to their agents, roles and parameters (terms) of specific actions. We propose to use unification to detect when a prohibition and a permission overlap and to employ the unifier to resolve the conflict. For instance, $\mathsf{P}_{A:R}p(c, X)$ and $\mathsf{F}_{a:b}p(Y, Z)$ are in conflict as they unify under $\sigma = \{A/a, R/b, Y/c, X/d\}$). If, however, the variables in $\mathsf{F}_{a:b}p(Y, Z)$ do not get the values in σ then there will be no conflicts. We thus propose to annotate the prohibitions in Ω with unifiers, called here *conflict sets*, and use these annotations to determine what the variables of the prohibition *cannot* be in future unifications in order to avoid a conflict. Each prohibition is henceforth regarded as having such an annotation, denoted as $\langle (\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e \rangle$. Initially, this annotation is empty.

We propose to *curtail* the influence of prohibitions, thus giving agents more choices in the actions they may perform. A similar approach could be taken whereby permissions are curtailed, thus limiting the available agents' actions. Each of these policies is possible: we do not legislate over any of them nor do we give preference over any. In this paper we are interested in formalising such policies within a simple mathematical framework. A prohibition can be in conflict with various permissions in Ω. We, therefore, have to find the *maximal* set of conflicting pairs of permissions and prohibitions in Ω, by performing a pairwise inspection. This requires identifying the substitution between two pairs of norms that characterises a conflict. This is formally captured by the following definition:

DEF. 7. *A conflict arises between two tuples $\omega, \omega' \in \Omega$ under a substitution σ, denoted as $\mathbf{cflct}(\omega, \omega', \sigma)$, iff the following conditions hold:*

1. $\omega = \langle (\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e \rangle$, $\omega' = \langle \mathsf{P}_{\tau'_1:\tau'_2}\varphi', t'_d, t'_a, t'_e \rangle$
2. $unify(\tau_1, \tau'_1, \sigma)$, $unify(\tau_2, \tau'_2, \sigma)$, *and* $unify(\varphi, \varphi', \sigma)$
3. $|t_e - t'_e| \leq |t_a - t'_a|$

That is, a prohibition and a permission conflict (condition 1) if, and only if, the agents and roles they apply to and their actions, respectively, unify under σ (condition 2) and their activation periods overlap (condition 3). Substitution σ, the conflict set, unifies the agents, roles and atomic formulae of a permission and a prohibition. The annotation Σ_c does not play any role when detecting conflicts, but, as we show below, we have to update the annotation to reflect new curtailments to solve conflicts. For instance, $\mathbf{cflct}(\langle (\mathsf{F}_{a:b}p(Y, d)) \odot \emptyset, 1, 3, 5 \rangle, \langle \mathsf{P}_{A:R}p(c, X), 2, 3, 4 \rangle, \{A/a, R/b, Y/c, Z/X\})$ holds. We define below how we obtain the set of conflicting norms of a normative state Ω:

DEF. 8. *The finite, possibly empty set of conflicting norms of a normative state Ω, denoted as $CFLS(\Omega)$, is defined as*

$$CFLS(\Omega) = \{\langle \omega, \omega', \sigma \rangle | \omega, \omega' \in \Omega, \mathbf{cflct}(\omega, \omega', \sigma)\}$$

3.2 Conflict Resolution

A fine-grained way of resolving conflict can be done via unification. We detect the overlapping of the norms' influences, *i.e.* how they affect the behaviours of agents in the VO, and we curtail the

influence of the prohibition. We illustrate with Venn diagrams in Figure 2 the overlap of norm influences (left) which characterises a conflict and the curtailment necessary to resolve the conflict (right). The illustration shows the space of possible values for $p(X, Y)$ and

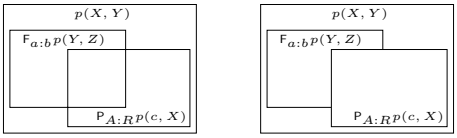

Figure 2: Overlap of Influence (Left) and Curtailment (Right)

two portions of this space defining the scope of influence of norms $\mathsf{P}_{A:R}p(c, X)$ and $\mathsf{F}_{a:b}p(Y, Z)$. The scope of these norms overlap, illustrated by the intersection of boxes on the left, in actions with values, for instance, $\langle a, b, p(c, 2)\rangle, \ldots, \langle a, b, p(c, n)\rangle$. The curtailment of the prohibition eliminates the intersection: it moves the scope of the norm influence to outside the influence of the permission. If there were multiple overlaps among one prohibition and various permissions, which is likely to happen, then the prohibition will be multiply curtailed to move the scope of the norm to avoid *all* intersections.

The algorithm shown in Figure 3 depicts how we obtain a conflict-free set of norms. It maps an existing set Ω possibly with conflicting norms onto a new set Ω' in which the conflicts (if any) are resolved. The algorithm forms Ω' as a set that is "conflict-free" – this means that prohibitions are annotated with a *conflict set* that indicates which bindings for variables have to be avoided.

algorithm $conflictResolution(\Omega, \Omega')$
input Ω
output Ω'
begin
  $\Omega' := \Omega$
  for each $\omega \in \Omega$ s.t. $\omega = \langle(\mathsf{F}_{a:r}\bar{\varphi}) \odot \Sigma_c, t_d, t_a, t_e\rangle$ **do**
    if $\langle \omega, \omega', \sigma\rangle \in CFLS(\Omega)$ **then** $\Omega' := \Omega' - \{\omega\}$
  end_for
  for each $\omega \in \Omega'$ s.t. $\omega = \langle(\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e\rangle$ **do**

$$\Sigma_c^{MAX} := \bigcup_{\langle \omega, \omega', \sigma'_c\rangle \in CFLS(\Omega')} \{\sigma'_c\}$$

$$\Omega' := (\Omega' - \{\omega\}) \cup \{\langle(\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot (\Sigma_c \cup \Sigma_c^{MAX}), t_d, t_a, t_e\rangle\}$$

  end_for
end

Figure 3: Algorithm to Resolve Conflicts in a Set of Norms

Initially, Ω' is set to be Ω. The algorithm operates in two stages. In the first stage (first **for each** loop), we remove all conflicting prohibitions $\omega = \langle(\mathsf{F}_{a:r}\bar{\varphi}) \odot \Sigma_c, t_d, t_a, t_e\rangle$ with ground agent/role pairs $a : r$ and ground formulae $\bar{\varphi}$: the only way to resolve conflicts arising from such prohibitions is to remove them altogether, as we cannot curtail a fully ground norm. In the second stage (second **for each** loop), the remaining prohibitions in Ω' are examined: the set $CFLS(\Omega')$ contains all conflicts between permissions and the remaining prohibitions in Ω' represented as tuples $\langle \omega, \omega', \sigma'_c\rangle$, with σ'_c representing the conflict set. As a prohibition may have conflicts with various permissions, the set $CFLS(\Omega')$ may contain more than one tuple for each prohibition. In order to provide an Ω' that reflects all these conflicts for a specific prohibition, we have to form Σ_c^{MAX} containing all conflict sets σ'_c for a given prohibition ω. The maximal set is used to update the annotation of the prohibition.

It is important to explain the need for updating the conflict set of prohibitions. Normative states change as a result of agents' actions [11]: existing permissions, prohibitions and obligations are revoked and/or new ones are put in place as a result of agents' interactions with the environment and other agents. Whenever new norms are added we must check for new conflicts and inconsistencies. If we only apply our algorithm to a pair consisting of an old and a new norm, then some re-processing of pairs of old norms (which were dealt with before) can be saved. The removal of norms from the set Ω is dealt with efficiently: each permission to be removed must be checked first for conflicts with any existing prohibition (re-processing can be avoided if we record the conflict, instead of detecting it again). If there is a conflict, then the conflict set will have been recorded in the prohibition's annotation; this conflict set is thus removed from the prohibition's annotation. The removal of obligations follows a similar process. Prohibitions are removed without the need to consider their relationships with other norms.

Our algorithm is correct in that it provides, for a given Ω, a new Ω' in which *i)* all ground prohibitions which conflict with permissions have been removed; and *ii)* all remaining annotated prohibitions $\langle(\mathsf{F}_{\tau:\tau'}\bar{\varphi}) \odot \Sigma_c, t_d, t_a, t_e\rangle$ will not unify with any of the permissions in Ω', provided the unifier does not appear in Σ_c. The first requirement is addressed by the first **for each** loop, which does precisely this: it removes all ground prohibitions which unify with an obligation. The second requirement is addressed by the second **for each** loop: each prohibition has its annotation Σ_c added with Σ_c^{MAX}, thus accommodating the unifiers from all permissions that unify with the prohibition. It is easy to see that the algorithm always terminates: each of its two loops go through a finite set, processing one element at a time. The set $CFLS(\Omega)$ is computed in a finite number of steps as are the set operations performed within each loop. The algorithm has, however, exponential complexity[3], as the computation of $CFLS(\Omega)$ requires a pairwise comparison of all elements in Ω.

We illustrate our algorithm with the following example. Let there be the following global normative state Ω:

$$\left\{ \begin{array}{ll} \langle(\mathsf{F}_{A:R}p(X, Y)) \odot \{\}, 2, 2, 9\rangle, & \langle\mathsf{P}_{a:r}p(a, b), 3, 4, 8\rangle, \\ \langle(\mathsf{F}_{a:r}p(a, b)) \odot \{\}, 2, 4, 12\rangle & \langle\mathsf{P}_{a:r}p(d, e), 3, 4, 9\rangle, \end{array} \right\}$$

The first loop removes the ground prohibition, thus obtaining the following Ω':

$$\left\{ \begin{array}{ll} \langle(\mathsf{F}_{A:R}p(X, Y)) \odot \{\}, 2, 2, 9\rangle, & \langle\mathsf{P}_{a:b}p(c, d), 3, 4, 8\rangle, \\ \langle\mathsf{P}_{e:f}p(g, h), 3, 4, 9\rangle & \end{array} \right\}$$

We then have the following set of conflicting norms $CFLS(\Omega')$:

$$\left\{ \left\langle \begin{array}{l} \langle(\mathsf{F}_{A:R}p(X, Y)) \odot \{\}, 2, 2, 9\rangle, \\ \langle\mathsf{P}_{a:b}p(c, d), 3, 4, 8\rangle, \\ \{A/a, R/b, X/c, Y/d\} \end{array} \right\rangle, \left\langle \begin{array}{l} \langle(\mathsf{F}_{A:R}p(X, Y)) \odot \{\}, 2, 2, 9\rangle, \\ \langle\mathsf{P}_{e:f}p(g, h), 3, 4, 9\rangle, \\ \{A/e, R/f, X/g, Y/h\} \end{array} \right\rangle \right\}$$

For each prohibition $\omega \in \Omega'$ we retrieve all elements from $\langle w, w', \sigma\rangle \in CFLS(\Omega')$ and collect their σ's in Σ_c^{MAX}. The final Ω' is thus:

$$\left\{ \begin{array}{l} \langle(\mathsf{F}_{A:R}p(X, Y)) \odot \left\{ \begin{array}{l} \{A/a, R/b, X/c, Y/d\} \\ \{A/e, R/f, X/g, Y/h\} \end{array} \right\}, 2, 2, 9\rangle, \\ \langle\mathsf{P}_{a:r}p(a, b), 3, 4, 8\rangle, \langle\mathsf{P}_{a:r}p(d, e), 3, 4, 9\rangle, \end{array} \right\}$$

The annotated set of conflict sets should be understood as a record of past unifications, which informs how prohibitions should be used in the future in order to avoid any conflicts with permissions. We show in Section 5.1 how annotations are used by norm-aware agents.

4. NORM INCONSISTENCIES

If a substitution σ can be found that unifies an obligation and a prohibition, then a situation of *norm inconsistency* occurs [13]. The obligation demands that an agent performs an action that is forbidden. We can reuse the machinery, introduced above for resolving conflicts between permissions and prohibitions, in order to *a)* detect and *b)* resolve such inconsistencies. With Definition 7, we

[3]The combinatorial effort is not necessary anymore if instead we maintain a set of norms conflict-free: each time a new norm is to be introduced then we compare it with the existing ones, thus making the maintenance process of linear complexity.

express the nature of a conflict between a prohibition and permission. Similarly, a situation of inconsistency can be defined reusing this definition and replacing the P deontic modality with O. We can reuse the machinery for conflict resolution, developed previously, for resolving inconsistency. The conflict resolution algorithm can be applied without change to accumulate a maximal conflict set Σ_c^{MAX} for each prohibition in Ω that unifies with obligations.

5. NORM-AWARE AGENT SOCIETIES

We now describe how our norm-regulated VOs give rise to norm-aware agent societies. We address open and heterogeneous MASs: we accommodate external agents by providing each of them with a corresponding *governor agent* [12]. This is a kind of "chaperon" that interacts with an external agent, and observes and reports on its behaviour. We show our architecture in Figure 4 below: a number

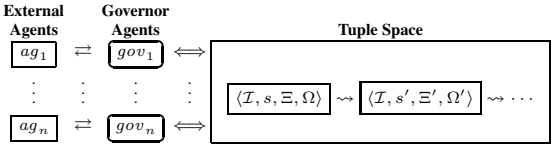

Figure 4: Architecture for Norm-Aware Agent Societies

of external agents interact (denoted by the "$\rightleftarrows$") with their corresponding governor agents. The governor agents have access to the VO description $\mathcal{I}$, the current state s of the VO enactment, the global execution state Ξ and the global normative state Ω. Governor agents are able to write to and read from (denoted by the "$\Longleftrightarrow$") a shared memory space (*e.g.*, a blackboard-like solution implemented as a tuple space), updating the global configuration (denoted by the "$\rightsquigarrow$") to reflect the dynamics of the VO enactment. Governor agents are necessary because we cannot anticipate or legislate over the design or behaviour of external agents. We depict below how the pairs of governor/external agents work together: any non-deterministic choices on the VO are decided by the external agent; any normative aspects are considered by the governor agent.

The governor agent represents the external agent within the VO. As such, it has the unique identifier of the external agent. The governor agent keeps an account of all roles the external agent is currently playing: in our VOs, it is possible for agents to take up more than one role simultaneously. We define in Figure 5 how governor agents work – we use a logic program for this purpose. We show

```
1  main(Id, Roles) ←
2    get_tuple(⟨I, s, Ξ, Ω⟩)∧
3    terminate(Id, Roles, I, Ξ, Ω)
4  main(Id, Roles) ←
5    get_tuple(⟨I, s, Ξ, Ω⟩)∧
6    filter_norms(Id, Roles, Ω, Ω_Id)∧
7    discuss_norms(Id, Roles, I, s, Ξ, Ω_Id, Actions)∧
8    update_tuple(Roles, Actions, NewRoles)∧
9    main(Id, NewRoles)
```

Figure 5: Governor Agent as a Logic Program

the lines of our clauses numbered 1-9. The first clause (lines 1-3) depicts the termination condition: `get_tuple`/1 (line 2) retrieves $\langle \mathcal{I}, s, \Xi, \Omega \rangle$ from the shared tuple space and `terminate`/4 checks if the current VO enactment (recorded in Ξ) has come to an end. The team of governor agents synchronise their access to the tuple space [12], thus ensuring each has a chance to function.

The second clause (lines 4-9) depicts a generic loop when the termination condition of the first clause does not hold. In this case, the tuple is again retrieved (line 5) and the governor agent proceeds (line 6) to analyse the current global normative state Ω with a view to obtaining the subset $\Omega_{Id} \subseteq \Omega$ of norms referring to agent Id under roles $Roles$. Predicate `filter_norms`/4 collects the norms which apply to agent Id (the governor agent's external agent). In line 7 the governor agent, in possession of the applicable norms as well as other relevant information, interacts with the external agent to decide on a set of $Actions$ which are norm-compliant – these actions will be used to update (line 8) the global execution state Ξ. In the process of updating the state of execution, a new set of roles must be assigned to the external agent, represented as $NewRoles$. The governor agent keeps looping (line 9) using the identifier for the external agent and its new set of roles.

5.1 Using Annotated Norms

We now explain how annotated norms are used by norm-aware agents. We do so via the definition of predicate `check`/2, which holds if its first argument, a candidate action (in the format of the elements of Ξ of Def. 2), is within the influence of an annotated prohibition ω, its second parameter. The definition, as a logic program, is shown in Figure 6. It checks (line 4) if the agent identifier

```
1  check(Action, ω) ←
2    Action = ⟨a : r, φ̄, t⟩∧
3    ω = ⟨(F_{τ1:τ2} φ) ⊙ Σ_c, t_d, t_a, t_e⟩∧
4    unify(a, τ1, σ) ∧ unify(r, τ2, σ) ∧ unify(φ̄, φ, σ)∧
5    forall(σ', (σ_c ∈ Σ_c, unify(σ_c, σ, σ')), MGUs)∧
6    MGUs = ∅∧
7    t_a ≤ t ≤ t_e
```

Figure 6: Check if Action is within Influence of Curtailed Norm

and role of the action unify with the appropriate terms τ_1, τ_2 of ω and that the actions $\bar{\varphi}, \varphi$ themselves unify, all under the same unifier σ. It then verifies (lines 5-6) that σ does *not* unify with any of the conflict sets in Σ_c. Finally, in line 7 it checks if the time of the action is within the norm temporal influence.

The verification of non-unification of σ with any element of Σ_c deserves an explanation. The elements of Σ_c are unifiers stating what values the variables of the norm cannot have, that is, they represent "gaps" in the original scope of the norm's influence. The test thus equates to asking if the action is outside such gaps, that is, the action is within the curtailed scope of influence of the norm.

6. ACTION CONFLICT & INCONSISTENCY

In our previous discussion, norm conflict and inconsistency were detected via a direct comparison of the atomic formulae representing the action. However, conflicts and inconsistencies may also arise *indirectly* via relationships among actions. For instance, if $p(X)$ amounts to $q(X, X)$, then norms $\mathsf{P}_{A:R}p(X)$ and $\mathsf{F}_{A:R}q(X, X)$ are in conflict since $\mathsf{P}_{A:R}p(X)$ can be rewritten as $\mathsf{P}_{A:R}q(X, X)$ and we thus have both $\mathsf{P}_{A:R}q(X, X)$ and $\mathsf{F}_{A:R}q(X, X)$. In the discussion below we concentrate on norm conflict, but norm inconsistency can be dealt with similarly, if we change the deontic modalities P for O.

Relationships among actions are domain-dependent. Different domains have distinct ways of relating their actions; engineers build ontologies to represent such relationships. We propose a simple means to account for such relationships and show how these can be connected to the mechanisms introduced above. Rather than making use of sophisticated formalisms for ontology construction, we employ a set of *domain axioms*, defined below:

DEF. 9. *The domain axioms, denoted as Δ, are a finite and possibly empty set of formulae $\varphi \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_n)$ where $\varphi, \varphi'_i, 1 \leq i \leq n$, are atomic first-order formulae.*

Our example above can be captured by $\Delta = \{(p(X) \rightarrow q(X, X)), (q(X, X) \rightarrow p(X))\}$. By explicitly representing and manipulating domain knowledge we achieve generality: the very same machinery can be used with different domains. A set of norms can have different conflicts and inconsistencies, for distinct domains of application.

We now revisit Def. 7 and augment it to account for action relationships. We do so via the logic program of Fig. 7, defining predicate $\mathbf{cflct}^*_\Delta(\Omega, \omega, \Omega', \sigma)$: it holds iff an indirect conflict arises between an annotated prohibition $\omega \in \Omega$ and a set of permissions $\Omega' \subseteq \Omega$ under a set of domain axioms Δ and a substitution σ. Clause 1 (lines 1-5) addresses the base case, when the action of the

1 $\quad \mathbf{cflct}^*_\Delta(\Omega, \omega, \Omega', \sigma) \leftarrow$
2 $\quad\quad \omega = \langle(\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e\rangle \wedge$
3 $\quad\quad \omega' \in \Omega \wedge \omega' = \langle \mathsf{P}_{\tau'_1:\tau'_2}\varphi', t'_d, t'_a, t'_e\rangle \wedge \Omega' = \{\omega'\} \wedge$
4 $\quad\quad unify(\tau_1, \tau'_1, \sigma) \wedge unify(\tau_2, \tau'_2, \sigma) \wedge unify(\varphi, \varphi', \sigma) \wedge$
5 $\quad\quad |t_e - t'_e| \leq |t_a - t'_a|$
6 $\quad \mathbf{cflct}^*_\Delta(\Omega, \omega, \Omega', \sigma) \leftarrow$
7 $\quad\quad \omega = \langle(\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e\rangle \wedge$
8 $\quad\quad (\varphi' \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_n)) \in \Delta$
9 $\quad\quad unify(\varphi, \varphi', \sigma) \wedge$
10 $\quad\quad \bigwedge_{i=1}^n \omega'_i = \langle((\mathsf{F}_{\tau_1:\tau_2}\varphi'_i) \cdot \sigma) \odot \Sigma_c, t_d, t_a, t_e\rangle$
11 $\quad\quad \bigwedge_{i=1}^n \mathbf{cflct}^*_\Delta(\Omega, \omega'_i, \Omega'_i, \sigma_i) \wedge$
12 $\quad\quad \Omega' = \bigcup_{i=1}^n \Omega'_i \wedge$
13 $\quad\quad \sigma = \bigcup_{i=1}^n \sigma_i$

Figure 7: Detection of Indirect Norm Conflicts

given annotated prohibition $\omega \in \Omega$ matches the action of a permission $\omega' \in \Omega$ under σ – the additional conditions on σ and time constraints are the same as those of Def. 7. Clause 2 (lines 6-12) addresses the general recursive case: if the prohibited action φ unifies (line 9) with φ' of a domain axiom $(\varphi' \rightarrow (\varphi'_1 \wedge \cdots \wedge \varphi'_n))) \in \Delta$ (line 8), then we build (line 10) new ω'_is with the φ'_i on the right-hand side of the axiom – notice that any unifications between φ and φ' are preserved on the φ'_is via substitution σ (applied to each φ'_i). In step 11 we recursively find the conflicting norms for *all* actions $\varphi'_1 \wedge \cdots \wedge \varphi'_n$ (line 10), assuming they are prohibited along the lines of φ. The final resulting Ω' is the union of the sets Ω_i recursively computed (line 12) and the final resulting σ is the union of the σ_i's (line 13).

It is interesting to notice that the domain axioms can be made very precise. For instance, if $p(a, X) \rightarrow q(a, f(a, X), X)$ is a domain axiom and we are dealing with prohibition $\mathsf{F}_{a:r}p(b, 2)$, then actions $p(b, 2)$ and $p(a, X)$ would not unify and an indirect norm conflict would not be possible. If, on the other hand, we had a prohibition $\mathsf{F}_{a:r}p(Y, Z)$, then we should be looking for an indirect conflict with prohibition $\mathsf{F}_{a:r}q(a, f(a, Z), Z)$.

6.1 Delegation between Roles

We can extend the set of domain axioms above to deal with conflicts and inconsistencies of norms when delegation of actions between roles takes place. For such situations, we introduce a special logical operator $\varphi \xrightarrow{\tau_1:\tau_2\ \tau'_1:\tau'_2} (\varphi'_1 \wedge \cdots \wedge \varphi'_n)$ to represent that agent τ_1 adopting role τ_2 can transfer any norms on action φ to agent τ'_1 adopting role τ'_2 which should carry out actions $\varphi'_1 \wedge \cdots \wedge \varphi'_n$ instead. We formally capture the meaning of this operator by adapting the 2nd clause of our logic program to detect indirect norm conflicts above, which will become as shown in Figure 8. In step 8 we obtain one domain axiom with an agent/role

6 $\quad \mathbf{cflct}^*_\Delta(\Omega, \omega, \Omega', \sigma) \leftarrow$
7 $\quad\quad \omega = \langle(\mathsf{F}_{\tau_1:\tau_2}\varphi) \odot \Sigma_c, t_d, t_a, t_e\rangle \wedge$
8 $\quad\quad (\varphi' \xrightarrow{\tau'_1:\tau'_2\ \tau''_1:\tau''_2} (\varphi'_1 \wedge \cdots \wedge \varphi'_n)) \in \Delta$
9 $\quad\quad unify(\varphi, \varphi', \sigma) \wedge unify(\tau_1, \tau'_1, \sigma) \wedge unify(\tau_2, \tau'_2, \sigma) \wedge$
10 $\quad\quad \bigwedge_{i=1}^n \omega'_i = \langle((\mathsf{F}_{\tau''_1:\tau''_2}\varphi'_i) \cdot \sigma) \odot \Sigma_c, t_d, t_a, t_e\rangle$
11 $\quad\quad \bigwedge_{i=1}^n \mathbf{cflct}^*_\Delta(\Omega, \omega'_i, \Omega'_i, \sigma_i) \wedge$
12 $\quad\quad \Omega' = \bigcup_{i=1}^n \Omega'_i \wedge$
13 $\quad\quad \sigma = \bigcup_{i=1}^n \sigma_i$

Figure 8: Detection of Conflicts caused by Delegation

delegation and on step 9 we check the axiom applies (*i.e.* unifies) with the prohibition ω. Step 10 "translates" ω using the axiom: not only new actions have to be checked, but these will be associated via norms to possibly different agent/role $\tau''_1 : \tau''_2$. Steps 11-13 are as before. For instance, if $\omega = \langle(\mathsf{F}_{A:r}p(X, 2)) \odot \Sigma_c, 20, 20, 40\rangle$ and $(p(Y, Y) \xrightarrow{a:R\ \ c:R} (q(Y) \wedge r(Y, Y))) \in \Delta$, then steps 7-9 above would yield $\omega'_1 = \langle(\mathsf{F}_{c:r}q(2)) \odot \Sigma_c, 20, 20, 40\rangle$ and $\omega'_2 = \langle(\mathsf{F}_{c:r}r(2, 2)) \odot \Sigma_c, 20, 20, 40\rangle$ – the transfer of prohibition would be specialised to the individual agent a adopting role r.

7. EXAMPLE: AGENTS FOR THE GRID

Service provision for e-Science using Grid infrastructure must be based on agreements to regulate the use of Grid services. When service consumers and providers engage, they have to establish agreements that regulate their interaction – specific obligations, prohibitions and rights have to be put in place in order to specify what the normative position of the partners in such a transaction will be. For example, a service provider will request payment that introduces a financial obligation on a user and, at the same time, gives the user rights to access the service for a certain period.

In order to illustrate these issues, we choose a scenario where the Principal Investigator (PI) of a research project has to analyse data as a specific research task. We assume that a contract exists between the PI and the funding body that introduces certain restrictions and obligations. We assume further that both PI and funding body are represented as agents operating on the Grid and that this contract is available in an electronic form that guide the behaviour of these agents. We assume that the PI (formalised as role pi) is represented by a special Research Support Agent (denoted as rsa) to which the PI delegates the task to perform the data analysis. We also assume that the Research Support Agent does not have the capability to perform the data analysis task and, therefore, has to outsource this activity. For this, it has to find a Grid service that can perform the required data analysis. We assume that the contract with the funding body states that (a) it is forbidden that project funds be used for outsourcing tasks and (b) that data used must not be disclosed. Elements of contract C are illustrated below, using the notation introduced previously (we assume t_d, t_a, t_e are, respectively, $1, 1, 1000$ throughout the example):

$$C = \left\{ \begin{array}{l} \langle \mathsf{F}_{rsa:pi}\ claim(X), 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(staff\_costs), 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(travel), 1, 1, 1000\rangle \\ \langle \mathsf{O}_{rsa:pi}\ report\_experiment(rsa, D), 1, 1, 1000\rangle \\ \langle \mathsf{F}_{X:Y}\ publish(D), 1, 1, 1000\rangle \end{array} \right\}$$

7.1 Resolving Norm Conflicts

The first three norms represent aspects of the financial requirements of the agent taking on the principal investigator role. All claims are prohibited (norm 1) with the exception of a number of specific types of item: staff costs (norm 2) and travel costs (norm 3) are itemised here. In addition, an obligation is stated that requires the PI to report about the experiment as well as a prohibition for anybody to publish data. The last norm is a basic prohibition, forbidding any agent in any role to publish data. The norms above are in conflict and we use our machinery to obtain a conflict-free version C' of it, in which only the first prohibition is curtailed:

$$C' = \left\{ \begin{array}{l} \langle(\mathsf{F}_{rsa:pi}\ claim(X)) \odot \left\{ \begin{array}{c} \{X/staff\_costs\}, \\ \{X/travel\} \end{array} \right\}, 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(staff\_costs), 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(travel), 1, 1, 1000\rangle \\ \langle \mathsf{O}_{rsa:pi}\ report\_experiment(rsa, D), 1, 1, 1000\rangle \\ \langle \mathsf{F}_{X:Y}\ publish(D), 1, 1, 1000\rangle \end{array} \right\}$$

7.2 Delegating Activities

In our example two Grid services are made available by two potential subcontractors for the execution of the data analysis task:

- a public non-profit organization provides a free service, but requires the disclosure of data in a public repository;
- a private commercial organization provides the service without the need for disclosure, but requests a payment.

These conditions of use can be expressed as norms in our formalism. The terms of the service, provided by the public non-profit organization, are:

$$N_1 = \{\langle \mathsf{O}_{A:R}\ publish(D'), 1, 1, 1000\rangle\}$$

That is, according to the terms of conditions of the public service, the input data have to be published. The terms of the service of the private commercial organization, on the other hand, are:

$$N_2 = \{\langle \mathsf{O}_{A:R}\ pay(fee), 1, 1, 1000\rangle\}$$

That is, whoever uses their service is obliged to pay a fee. The Research Assistant Agent has to choose which service to use. Each case introduces a new obligation with inconsistencies, explained below.

7.3 Inconsistency Resolution

If the public Grid service is chosen, then the set N_1, containing a new obligation, is introduced. The set $C' \cup N_1$ contains an inconsistency: the obligation to publish overlaps with the influence of the prohibition to publish. Our machinery copes with this, completely curtailing the prohibition and giving rise to a new set C'':

$$C'' = \begin{Bmatrix} \langle(\mathsf{F}_{rsa:pi}\ claim(X)) \odot \begin{Bmatrix}\{X/staff\_costs\},\\ \{X/travel\}\end{Bmatrix}, 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(staff\_costs), 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(travel), 1, 1, 1000\rangle \\ \langle \mathsf{O}_{rsa:pi}\ report\_experiment(rsa, D), 1, 1, 1000\rangle \\ \langle(\mathsf{F}_{X:Y}\ publish(D)) \odot \{\{D/D'\}\}, 1, 1, 1000\rangle \end{Bmatrix}$$

The pair D/D' expresses that variable D cannot be bound to anything (since D' is a free variable).

7.4 Indirect Inconsistency Resolution

If the private service is chosen, then the set N_2 is introduced. By forming the set $C' \cup N_2$, we introduce an *indirect* inconsistency. Set C' expresses a general prohibition to claim expenses, curtailed by a set of specific permissions that allow the spending of money for staff and travel. Intuitively, we know that an obligation to pay a fee for a service generates an inconsistency, because the original contract does not allow such a claim. This can be represented with the domain axiom

$$\Delta = \{pay(X) \xrightarrow{A:R\ \ A:R} claim(X)\}$$

That is, the axiom states that to pay for something amounts to claiming it. By applying the indirect conflict/inconsistency detection mechanism of Figure 8, we extend the annotation of the prohibition in the contract, thus further curtailing it and producing a C^3:

$$C^3 = \begin{Bmatrix} \langle(\mathsf{F}_{rsa:pi}\ claim(X)) \odot \begin{Bmatrix}\{X/staff\_costs\},\\ \{X/travel\},\\ \{X/fee\}\end{Bmatrix}, 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(staff\_costs), 1, 1, 1000\rangle \\ \langle \mathsf{P}_{rsa:pi}\ claim(travel), 1, 1, 1000\rangle \\ \langle \mathsf{O}_{rsa:pi}\ report\_experiment(rsa, D), 1, 1, 1000\rangle \\ \langle(\mathsf{F}_{X:Y}\ publish(D)) \odot \{\{D/d_1\}\}, 1, 1, 1000\rangle \end{Bmatrix}$$

It is interesting to notice that the annotation on the first norm above solves conflicts as well as inconsistencies.

7.5 Solving Conflicts arising from Delegation

Our example also exploits conflicts arising from delegation among agents/roles. Let there be the following domain axioms:

$$\Delta = \begin{Bmatrix} pay(X) \xrightarrow{A:R\ \ A:R} claim(X) \\ report\_experiment(A, E, D) \xrightarrow{A:R\ \ A:R} do\_exp(A, E, D) \\ do\_exp(A, e_1, D) \xrightarrow{A:pi\ \ A:pi} send(A, exp, e_1, D) \\ send(A, R', E, D) \xrightarrow{A:R\ \ A':R'} receive(A', R', A, E, D) \\ receive(A', R', A, E, D) \xrightarrow{A':R'\ \ A':R'} \begin{pmatrix} analyse(A', E, D, S) \wedge \\ send(A, A', S) \end{pmatrix} \end{Bmatrix}$$

The set Δ contains axioms that describe how the Research Assistant Agent can fulfil its obligation to report the result of an experiment. As the domain axioms show, there is a relationship between the action $report\_experiment$ and do_exp. An additional axiom tells us that the action do_exp leads to the sending of experimental data to one of the chosen Grid services of subcontractors. The domain axiom

$$send(A, R', E, D) \xrightarrow{A:R\ \ A':R'} receive(A', R', A, E, D)$$

shows the delegation of activities from the agent responsible for the data analysis to a subcontractor for actually performing the experiment. The rest of the domain axioms describe how a subcontractor performs an experiment and sends back results upon receiving such a request. According to the detection process depicted in Figure 8, the obligation to perform a specific task is transferred to related actions via the domain axioms. For example, the obligation to report experimental results gives rise to an obligation to perform the action do_exp and, continuing in this transitive fashion, obligations for all the related actions as described before. Due to the delegation step, obligations also arise for the partner agents. These obligations, in their turn, may interfere with prohibitions held by the collaborating agents and may have to be dealt with in the same way.

8. RELATED WORK

Socio-philosophical studies on norms and agents highlight the importance of norms in agent behaviour, *e.g.*, [5] and [26], or analyse the emergence of norms in multi-agent systems, *e.g.*, [27] and [25]. On the other hand, logic-theoretic contributions focus on the deontic logics required to model normative modalities along with their paradoxes, *e.g.*, [6] and [24]. The last few years, however, have seen significant work on norms in MASs.

Formal attributes of legal systems such as consistency and generality are specific interests in legal philosophy and legal theory. These formal attributes are studied in the context of jurisprudence and are also a concern in the application of computer science and artificial intelligence in the law-making process [7, 23]. Inconsistency in law is an important issue and legal theorists use a diverse set of terms such as, for example, normative inconsistencies/conflicts, antinomies, discordance, etc., in order to describe this phenomenon.

There are three classic strategies for resolving deontic conflicts: *legis posterior* (the most recent norm takes precedence), *legis superior* (the norm imposed by the strongest power takes precedence) [15], and *legis specialis* (the most specific norm takes precedence). We notice that our approach to resolving norm conflict and inconsistency can be combined with any of these. Additionally, *legis posterior* and *specialis* can be operationalised in our approach via the explicit manipulation of conflict sets – *legis posterior* for instance, can be implemented by distributing the conflict sets between both norms (reflecting their chronology).

In [10] the authors give algorithms for the classic strategies to resolve conflicts among normative positions of agents. That work also addresses interdependence among actions (named "compound activities"). It is not clear what the complexity of their algorithms is; additionally, the algorithms make use of an information model that some may regard as complex. The work presented in [13] discusses these kinds of strategies as well, also proposing conflict resolution via negotiation with a norm issuer. Our proposed mechanism for conflict resolution of norms has been sketched in [13], but only informally, using instantiation graphs – their high computational complexity only allow simple scenarios to be addressed. Additionally, that work only contemplates individual norm-compliant agents, and not an organizational infrastructure for open MASs

In [7] we find an analysis of different normative conflicts (albeit an informal one, in spite of its title) 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 real conflict actually occurs. The situations when an action is simultaneously obliged and prohibited are, however, deontic conflicts, as both obligations and prohibitions influence behaviours in a conflicting fashion. We notice that our approach to detecting deontic conflict/inconsistency can capture the three forms of conflict/inconsistency of [21], *viz.* total-total, total-partial and intersection: these are special cases of the intersection of Figure 2, respectively, when the permission entails the prohibition, when the prohibition entails the permission and when they simply overlap. Finally, we notice that the *world knowledge* explained in [7], required to relate actions, can be formally captured by our indirect norm conflicts depicted in Section 6.

The curtailment of norms can be related with work on default reasoning [22]: the interpretation of the defeasible inference operator $A \Rightarrow B$, that is, "if A holds and it can be consistently assumed that B holds, then B holds" is precisely what we achieve with our conflict sets, but in a fine-grained fashion. More formally, $\nu \odot \Sigma_c \Rightarrow \nu \cdot \sigma$ iff σ does not unify with any of the elements in Σ_c: individual values for variables are exceptions to the rule.

9. CONCLUSIONS & FUTURE WORK

We have introduced a fine-grained approach to resolving norm conflict (*i.e.*, when an action is simultaneously prohibited and permitted) and inconsistencies (*i.e.*, when an action is simultaneously prohibited and obliged) in norm-regulated virtual organizations (VOs). We addressed a generic (albeit simple) finite-state based formulation of VOs and complemented this model with an explicit account of the normative positions of agents taking part on the enactment of a VO, thus defining an expressive class of norm-regulated multi-agent systems. We have presented an algorithm (with auxiliary definitions) to examine a set of norms and obtain a conflict- and inconsistency-free version of it, and have discussed our algorithm's termination, correctness and complexity. We can say our approach is fine-grained as it allows conflicts and inconsistencies to be resolved down to the level of individual agents and roles and particular values of actions' parameters.

Our proposal hinges on first-order unification, a fundamental feature of automatic theorem proving. Although we do not commit ourselves to any particular unification algorithm, we do rely on the termination, correctness and linear complexity of such procedures. Our proposal uses unification to precisely detect if and when norms conflict or become inconsistent; we use the substitution which unifies the conflicting (or inconsistent) norms as annotations, thus saving the effort to compute the complement set of values variables may have in order to avoid the conflict (or inconsistency). We also proposed an architecture to utilise the annotated norms, thus endowing a society of agents with norm-awareness. Our approach neatly accommodates domain axioms relating actions: as a result we can detect indirect norm conflicts and inconsistencies due to actions being mapped to (conjunctions of) other actions. We can also account for delegation of actions among agents and roles, and via this we can capture delegation of norms, and their potential conflicts and inconsistencies.

We would like to extend our approach to cope with arbitrary constraints associated to actions. These constraints restrict the possible values for variables, thus conferring more precision, expressiveness and realism to norms. We would like to address, for instance, a prohibition such as "agent A adopting role R can perform action $p(X, Y)$, provided $X > Y + 20$". When such arbitrary constraints become part of norms, the unification (and hence conflict and inconsistency detection) must check if they overlap, that is, if they admit one possible solution. The curtailment could be reinstated as a manipulation of constraints.

Acknowledgements: This research is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence.

10. REFERENCES

[1] K. R. Apt. *From Logic Programming to Prolog*. Prentice-Hall, U.K., 1997.
[2] A. Artikis. *Executable Specification of Open Norm-Governed Computational Systems*. PhD thesis, Department of Electrical & Electronic Engineering, Imperial College London, Nov. 2003.
[3] A. Artikis, L. Kamara, J. Pitt, and M. Sergot. A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. volume 3476 of *LNCS*. Springer-Verlag, 2005.
[4] J. Broersen, F. Dignum, V. Dignum, and J.-J. C. Meyer. Designing a Deontic Logic of Deadlines. volume 3065 of *LNAI*. Springer-Verlag, 2004.
[5] R. Conte and C. Castelfranchi. Understanding the Functions of Norms in Social Groups through Simulation. In N. Gilbert and R. Conte, editors, *Artificial Societies: The Computer Simulation of Social Life*, pages 252–267, London, 1995. UCL Press.
[6] F. Dignum. Autonomous Agents with Norms. *AI & Law*, 7(1):69–79, 1999.
[7] A. Elhag, J. Breuker, and P. Brouwer. On the Formal Analysis of Normative Conflicts. *Information & Comms. Techn. Law*, 9(3):207–217, Oct. 2000.
[8] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, U.S.A., 1990.
[9] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int' J. Supercomputer Applications*, 15(3):209–235, 2001.
[10] A. García-Camino, P. Noriega, and J.-A. Rodríguez-Aguilar. An Algorithm for Conflict Resolution in Regulated Compound Activities. In *Seventh Annual International Workshop Engineering Societies in the Agents World (ESAW'06)*, September 2006.
[11] A. García-Camino, J.-A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. A Rule-based Approach to Norm-Oriented Programming of Electronic Institutions. *ACM SIGecom Exchanges*, 5(5):33–40, Jan. 2006.
[12] A. Garcia-Camino, J.-A. Rodriguez-Aguilar, C. Sierra, and W. W. Vasconcelos. A Distributed Architecture for Norm-Aware Agent Societies. volume 3904 of *LNAI*. Springer-Verlag, 2005.
[13] M. Kollingbaum, T. Norman, A. Preece, and D. Sleeman. Norm Refinement: Informing the Re-negotiation of Contracts. In G. Boella, O. Boissier, E. Matson, and J. Vazquez-Salceda, editors, *ECAI 2006 Workshop on Coordination, Organization, Institutions and Norms in Agent Systems, COIN@ECAI 2006*, pages 46–51, 2006.
[14] Y.-J. Lee. A Dynamic Virtual Organization Solution for Web-Services Based Grid Middleware. In *16th Int'l Workshop on Database and Expert Systems Applications (DEXA)*. IEEE Computer Society, 2005.
[15] J. A. Leite, J. J. Alferes, and L. M. Pereira. Multi-Dimensional Dynamic Knowledge Representation. volume 2173 of *LNAI*. Springer-Verlag, 2001.
[16] T. Norman, A. Preece, S. Chalmers, N. Jennings, M. Luck, V. Dang, T. Nguyen, V. Deora, J. Shao, W. Gray, and N. Fiddian. Agent-based Formation of Virtual Organisations. *Knowledge Based Systems*, 17:103–111, 2004.
[17] D. E. O'Leary, D. Kuokka, and R. Plant. Artificial Intelligence and Virtual Organizations. *Commun. ACM*, 40(1), 1997.
[18] O. Pacheco and J. Carmo. A Role Based Model for the Normative Specification of Organized Collective Agency and Agents Interaction. *Autonomous Agents and Multi-Agent Systems*, 6(2):145–184, Mar. 2003.
[19] H. V. D. Parunak and J. Odell. Representing Social Structures in UML. In *Procs 5th Int'l Conf. on Autonomous Agents*, pages 100–101, Montreal, Canada, 2001. ACM Press.
[20] J. A. Rodríguez-Aguilar. *On the Design and Construction of Agent-mediated Electronic Institutions*. PhD thesis, IIIA-CSIC, Spain, 2001.
[21] A. Ross. *On Law and Justice*. Stevens & Sons, 1958.
[22] L. Royakkers and F. Dignum. Defeasible Reasoning with Legal Rules. In D. Nute, editor, *Defeasible Deontic Logic*. Kluwer, 1997.
[23] G. Sartor. Normative Conflicts in Legal Reasoning. *AI & Law*, 1(2-3):209–235, June 1992.
[24] M. Sergot. A Computational Theory of Normative Positions. *ACM Trans. Comput. Logic*, 2(4):581–622, 2001.
[25] Y. Shoham and M. Tennenholtz. On Social Laws for Artificial Agent Societies: Off-line Design. *Artificial Intelligence*, 73(1-2):231–252, 1995.
[26] R. Tuomela and M. Bonnevier-Tuomela. Norms and Agreement. *European Journal of Law, Philosophy and Computer Science*, 5:41–46, 1995.
[27] A. Walker and M. Wooldridge. Understanding the Emergence of Conventions in Multi-agent Systems. In *Procs. Int'l Joint Conf. on Multi-Agent Systems (ICMAS)*, pages 384–389, San Francisco, USA, 1995.