Conflict Resolution in Norm-Regulated Environments via Unification and Constraints

M. J. Kollingbaum[1,*], W. W. Vasconcelos[1,†], A. García-Camino[2,◇], and T. J. Norman[1,‡]

[1]Dept. of Computing Science, Univ. of Aberdeen, Aberdeen AB24 3UE, UK
{*mkolling,† wvasconc,‡ tnorman}@csd.abdn.ac.uk
[2]IIIA-CSIC, Campus UAB 08193 Bellaterra, Spain
◇andres@iiia.csic.es

Abstract. We present a novel mechanism for the detection and resolution of conflicts within norm-regulated virtual environments, populated by agents whose behaviours are regulated by explicit obligations, permissions and prohibitions. A conflict between norms arises when an action is simultaneously prohibited and obliged or prohibited and permitted. In this paper, we use first-order unification and constraint satisfaction to detect and resolve such conflicts, introducing a concept of norm curtailment. A flexible and robust algorithm for norm adoption is presented and aspects of indirect conflicts and conflicts across delegation of actions between agents is discussed.

1 Introduction

Norm-governed virtual organisations use obligations, permissions and prohibitions for the regulation of the behaviour of self-interested, heterogeneous software agents. Norms are important in the design and management of virtual organisations, as they allow a detailed specification of these social structures in terms of roles and the rights and duties of agents adopting these roles. Norm-regulated VOs, however, may experience problems when norms assigned to agents are in *conflict* – actions that are forbidden, may, at the same time, also be obliged and/or permitted. For example, a norm "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 and Z are variables and ag_1, ag_2 and 20 are constants) show two norms that are in conflict regarding an action *send_bid*.

In order to detect and resolve norm conflicts and to check norm-compliance of actions, we propose a mechanism based on first-order term unification [1] and constraint satisfaction. With that, we develop further the work presented in [2] where we used first-order term unification for conflict detection and norm annotations to avoid conflicts indicating what the variables of a prohibition *cannot* be when actions are deployed. In this paper, we also use unification, but add constraint satisfaction for conflict detection and resolution.

In the following section, we introduce a "lightweight" definition of virtual organisations and their enactments. In Section 3 we define norms, constraints and global normative states. Section 4 describes in detail a machinery for conflict detection and resolution. In section 5, we describe how agents check the norm-compliance of their actions with the use of unification and constraint satisfaction. Section 6 describes *indirect* conflicts occurring via domain-specific relationships between actions and via the delegation between roles. Section 7 describes the

application of the conflict resolution machinery in a detailed example. Section 8 provides an overview about related work and section 9 concludes this paper.

2 Virtual Organisations

Following [2], we base our discussion of norm conflicts on a simple representation of a virtual organisation [3] as a finite-state machine where actions of individual agents lead to state transitions. Figure 1 depicts a graphical representation of

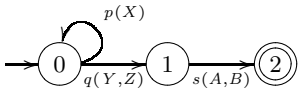

Fig. 1: Sample VO as a Finite-State Machine

this finite-state machine, where the edges between discrete states are labelled with first-order formulae representing actions performed by individual agents [1]. Although there are more sophisticated and expressive ways to represent agent activity and interaction (*e.g.*, AUML [5] and electronic institutions [6], 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). A virtual organisation is defined as follows:

Definition 1. *A virtual organisation $\mathcal{I}$ is the tuple $\langle S, s_0, E, T\rangle$, 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', φ) with $s, s' \in S$ connecting s to s' and labelled with a first-order atomic formula φ, 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 $\exists\, A, B\ s(A, B)$.

Roles, as exploited in, for instance, [7] and [6], 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.

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

[1] We adopt Prolog's convention [4] 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 [8] and is explained in section 5.

a set of ground atomic formulae $\bar{\varphi}$, 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:

Definition 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 a 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, $h^*(\mathcal{I}, \Xi, s) = \{\Xi \cup \{\langle a{:}r, \bar{\varphi}, t\rangle\} \mid (s, s', \varphi) \in E\}$.

The VO specification introduced previously must be augmented to accommodate the agent identification as well as its associated role. We thus have edges specified as $(s, s', \langle a, r, \varphi, t\rangle)$. More expressiveness can be achieved if we allow constraints (as introduced below) to be added to edges, as in, for instance, $(s, s', \langle a, r, (p(X, Y) \wedge X > Y), t\rangle)$, depicting that the formula $p(X, Y)$ causes the progress of the VO, provided $X > Y$. Such VOs are as expressive as the logic-based electronic institutions proposed in [9].

3 Norms

Norms are the central element in our discussion. We regard agents adopting specific roles and, with that, a set of norms that regulate their actions within a virtual organisation. We extend our previous work [2], and introduce a more expressive norm definition, accommodating constraints. We, again, adopt the notation of [10] for specifying norms and complement it with *constraints* [11]. By using constraints, we can *restrict* the influence of norms on specific parameters of actions. Our building blocks are first-order terms τ, that is, constants, variables and functions (applied to terms). We shall make use of numbers and arithmetic functions to build those terms. Arithmetic functions may appear infix, following their usual conventions. Constraints are defined as follows:

Definition 3. *Constraints, generically represented as γ, are any construct of the form $\tau \lhd \tau'$, where $\lhd \in \{=, \neq, >, \geq, <, \leq\}$.*

We then introduce the syntax of norms:

Definition 4. *A norm ω is a tuple $\langle \nu, t_d, t_a, t_e\rangle$, where ν is any construct of the form $\mathsf{O}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i$ (an obligation), $\mathsf{P}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i$ (a permission) or $\mathsf{F}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i$ (a prohibition), where τ_1, τ_2 are terms, φ is a first-order atomic formula and γ_i, $0 \leq i \leq n$, are constraints. The elements $t_d, t_a, t_e \in \mathbb{N}$ are, respectively, the time when ν was declared (introduced), when ν becomes active and when ν expires, $t_d \leq t_a \leq t_e$.*

Term τ_1 identifies the agent(s) to whom the norm is applicable and τ_2 is the role of such agent(s). $\mathsf{O}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i$ thus represents an obligation on agent τ_1 taking up role τ_2 to bring about φ, subject to constraints γ_i, $0 \leq i \leq n$. The γ_i's express constraints on those variables occurring in φ.

In the definition above, we only cater for conjunctions of constraints. If disjunctions are required then a norm must be established for each disjunct. For instance, if we required the norm $\mathsf{P}_{A:R}move(X) \wedge (X < 10 \vee X = 15)$ then we must break it into two norms $\mathsf{P}_{A:R}move(X) \wedge X < 10$ and $\mathsf{P}_{A:R}move(X) \wedge X = 15$. This holds because we assume an implicit universal quantification over variables in ν. For instance, $\mathsf{P}_{A:R}p(X, b, c)$ stands for $\forall\, A \in Agents. \forall\, R \in Roles. \forall\, X.\mathsf{P}_{A:R}p(X, b, c)$. We comment on the existential quantification in the final section of this paper.

We propose to formally represent the normative positions of all agents, taking part in a virtual society, from a global perspective. By "normative position" we mean the "social burden" associated with individuals [8], that is, their obligations, permissions and prohibitions:

Definition 5. *A global normative state Ω is a finite and possibly empty set of tuples $\omega = \langle \nu, t_d, t_a, t_e \rangle$.*

As a simplification, we assume a single global normative state Ω for a virtual organisation. However, this can be further developed into a fully distributed form, with each agent maintaining its own Ω, thus allowing the scaling up of our machinery.

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 be interested 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, $f(\Xi, \Omega) = \Omega - Obls$, $Obls = \{\langle \mathsf{O}_{a:r}\varphi, t_d, t_a, t_e\rangle \in \Omega \mid \langle a : r, \bar{\varphi}, t\rangle \in \Xi\}$. The management (*i.e.*, creation and updating) of global normative states is an interesting area of research. A simple and useful approach is reported in [12]: 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 proposing how Ω's should be managed, and assume some mechanism which does it.

4 Norm Conflicts

A conflict between two norms occurs if a formula representing an action is simultaneously *under the influence* of a permission and prohibition or an obligation and prohibition for the same agent (or set of agents) – the agent experiences a

normatively ambiguous situation for a specific set of actions. A norm *influences* who (what agent/set of agents in a specific role) is either permitted, prohibited or obliged to perform a specific action (or set of actions). We regard norms having a *scope of influence* as they may have an influence on a set of actions.

Figure 2 shows the scope of influence of a prohibition and a permission on instantiations of the action $shift(X, Y, Z), X \in \{a, b\}, Y \in \{r, s\}, Z \in \{u, v\}$,

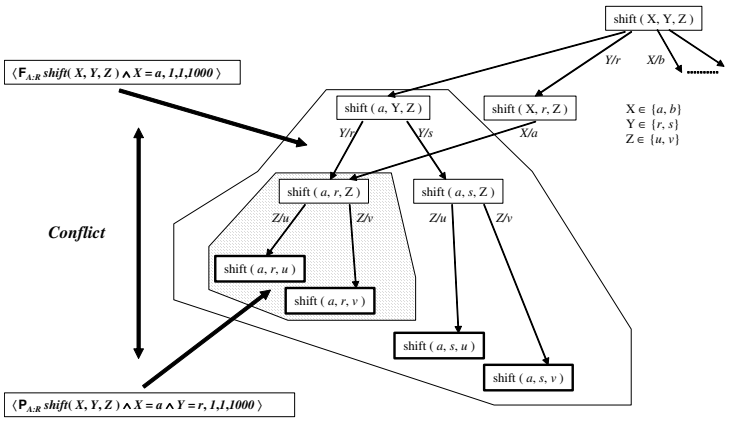

Fig. 2: Conflict between a Permission and a Prohibition

in a blocks world scenario, representing that block X is shifted from the top of block Y to the top of block Z. The prohibition prevents any agent in any role to shift a specific block a from any block to any block. The scope of this prohibition is the portion of the action's space of possibilities enclosed within the larger irregular polygon. The diagram also shows the scope of a permission conflicting with this prohibition – it permits any agent in any role to shift a specific block a from a specific block r to any other block. The scope of influence of the permission is the portion of $shift$'s space of possibilities enclosed within the smaller grey irregular polygon, contained within the scope of the prohibition. This is a typical situation of conflict – the scopes of influence of both norms overlap.

We use *unification* of first-order terms [4, 1] as an instrument to detect and resolve conflicts between norms. Unification allows us *i)* to detect whether norms are in conflict and *ii)* to detect the set of actions that are under the influence of a norm. Unification is a fundamental problem in automated theorem proving and many algorithms have been proposed [1], recent work proposing means to obtain unifiers efficiently. Unification is based on the concept of substitution:

Definition 6. *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 in accordance with [1] – a substitution σ is a *unifier* of two terms $\tau_1 : \tau_2$, if $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$. In addition, we describe, how substitutions are applied to obligations, permissions and prohibitions. Below, X stands for either O, P or F:

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. $(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i) \cdot \sigma = (\mathsf{X}_{(\tau_1\cdot\sigma):(\tau_2\cdot\sigma)}\varphi \cdot \sigma) \wedge \bigwedge_{i=0}^{n} \gamma_i \cdot \sigma)$.
5. $\langle \nu, t_d, t_a, t_e \rangle \cdot \sigma = \langle (\nu \cdot \sigma), t_d, t_a, t_e \rangle$

We shall use unification in the following way:

Definition 7. *$unify(\tau_1, \tau_2, \sigma)$ holds for two terms τ_1, τ_2, iff $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$ holds, for some σ; $unify(p^n(\tau_0, \ldots, \tau_n), p^n(\tau'_0, \ldots, \tau'_n), \sigma)$ holds, for two atomic formulae $p^n(\tau_0, \ldots, \tau_n), p^n(\tau'_0, \ldots, \tau'_n)$, iff $unify(\tau_i, \tau'_i, \sigma), 0 \leq i \leq n$, for 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 norms, we can use unification for detecting a conflict between two norms (X, X', again, stand for either O, P or F):

Definition 8. *$unify(\omega, \omega')$ holds for two norms $\omega = \langle (\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i), T_a, T_d, T_e \rangle$ and $\omega' = \langle (\mathsf{X}'_{\tau'_1:\tau'_2}\varphi' \wedge \bigwedge_{j=0}^{m} \gamma'_j), T'_a, T'_d, T'_e \rangle$, iff*

1. *$unify(\langle \tau_1, \tau_2, \varphi, T_a, T_d, T_e \rangle, \langle \tau'_1, \tau'_2, \varphi', T'_a, T'_d, T'_e \rangle, \sigma)$ and*
2. *$satisfy((\bigwedge_{i=0}^{n}(\gamma_i \cdot \sigma)) \wedge (\bigwedge_{j=0}^{m}(\gamma'_j \cdot \sigma)))$*

Two conditions are tested: the first one checks that the various components of a norm, organised as a tuple, unify; the second one checks that the constraints associated with the norms are satisfiable[3].

4.1 Conflict Detection

With unification, we can detect whether norms are in conflict. We define formally a conflict between norms as follows:

Definition 9. *A conflict arises between $\omega, \omega' \in \Omega$ under a substitution σ, denoted as* **conflict**$(\omega, \omega', \sigma)$*, iff the following conditions hold:*

1. *$\omega = \langle (\mathsf{F}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n} \gamma_i), t_d, t_a, t_e \rangle$, $\omega' = \langle (\mathsf{O}_{\tau'_1:\tau'_2}\varphi' \wedge \bigwedge_{i=0}^{n} \gamma'_i), t'_d, t'_a, t'_e \rangle$,*
2. *$unify(\langle \tau_1, \tau_2, \varphi \rangle, \langle \tau'_1, \tau'_2, \varphi' \rangle, \sigma)$, $satisfy(\bigwedge_{i=0}^{n} \gamma_i \wedge (\bigwedge_{i=0}^{m} \gamma'_i \cdot \sigma))$*
3. *$overlap(t_a, t_e, t'_a, t'_e)$.*

That is, a conflict occurs if *i)* a substitution σ can be found that unifies the variables of two norms[4], and *ii)* the conjunction $\bigwedge_{i=0}^{n} \gamma_i \wedge (\bigwedge_{i=0}^{m} \gamma'_i) \cdot \sigma)$ of constraints from both norms can be satisfied (taking σ under consideration), and *iii)* the activation period of the norms overlap. The *overlap* relationship holds if *i)* $t_a \leq t'_a \leq t_e$; or *ii)* $t'_a \leq t_a \leq t'_e$. For instance, for the two norms $\mathsf{P}_{A:R}p(c, X) \wedge X > 50$ and $\mathsf{F}_{a:b}p(Y, Z) \wedge Z < 100$, a substitution $\sigma = \{A/a, R/b, Y/c, X/Z\}$ can be found that allows the unification of both norms – being able to construct such a

[3] We assume an implementation of the *satisfy* relationship based on "off the shelf" constraint satisfaction libraries such as those provided by SICStus Prolog [13–15] and it holds if the conjunction of constraints is satisfiable.

[4] A similar definition is required to address the case of conflict between a prohibition and a permission – the first condition should be changed to $\omega' = \langle (\mathsf{P}_{\tau'_1:\tau'_2}\varphi' \wedge \bigwedge_{i=0}^{n} \gamma'_i), t'_d, t'_a, t'_e \rangle$. The rest of the definition remains the same.

unifier is a first indication that there may be a conflict, expressed as an overlap of their influence on actions. The unifier expresses that the two norms conflict if the variables A, R, Y and X receive as bindings the values contained in the unifier. On the other hand, there will be no conflict if different bindings are chosen. The constraints on the norms may restrict this overlap and, therefore, leave actions under certain variable bindings free of conflict. The constraints of both norms have to be investigated to see if an overlap of the values indeed occurs. In our example, the permission has a constraint $X > 50$ and the prohibition has $Z < 100$. By using the substitution X/Z, we see that $50 < X < 100$ and $50 < Z < 100$ represent ranges of values for variables X and Z where a conflict will occur.

For convenience (and without any loss of generality) we assume that our norms are in a special format: any non-variable term τ occurring in ν is replaced by a fresh variable X (not occurring anywhere in ν) and a constraint $X = \tau$ is added to ν. This transformation can be easily automated by scanning ν from left to right, collecting all non-variable terms $\{\tau_1, \ldots, \tau_n\}$; then we add $\wedge_{i=1}^{n} X_i = \tau_i$ to ν. For example, norm $\mathsf{P}_{A:R}p(c, X) \wedge X > 50$ is transformed into $\mathsf{P}_{A:R}p(C, X) \wedge X > 50 \wedge C = c$.

4.2 Conflict Resolution

In order to resolve a conflict with respect to a specific action that is located in the overlap of the scopes of influence of both norms, a social entity has to decide which of the two conflicting norms it should adhere and which it should ignore. For a software agent, a machinery has to be put in place that computes a possible disambiguation of its normative situation – the set of norms Ω has to be transformed into a set Ω' that does not contain any conflicting norms so that the agent can proceed with its execution. In [2], we achieved this by using a concept of *curtailment* – one of the norms is changed in a way so that its scope of influence is retracted from specific actions (which norm to choose for curtailment is a different matter and not discussed in this paper). By curtailing the scope of influence of a norm, the overlap between the two norms is eliminated.

Extending [2], we achieve curtailment by manipulating the constraints of the norms. In figure 3, we show how a curtailment of the prohibition changes its scope

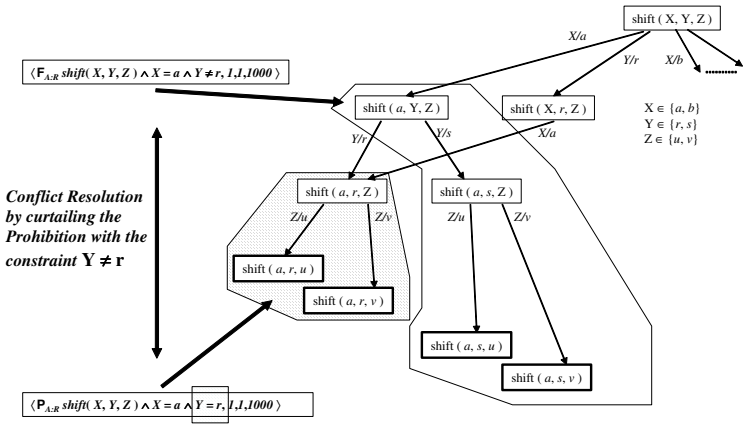

Fig. 3: Conflict Resolution with Curtailment

of influence and thus eliminates the overlap between the two norms. Specific constraints are added to the prohibition in order to perform this curtailment – as shown in figure 3, these additional constraints are derived from the permission. The scope of the permission is determined by the constraints $X = a$ and $Y = r$, restricting the set of bindings for variables X and Y to values a and r. Adding a constraint $Y \neq r$ to the prohibition curtails its scope of influence and eliminates the overlap with the scope of influence of the permission.

We now formally define how the curtailment of norms takes place. It is important to notice that the curtailment of a norm creates a new (possibly empty) set of curtailed norms:

Definition 10. *Relationship* **curtail**$(\omega, \omega', \Omega)$*, where* $\omega = \langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i, t_d, t_a, t_e\rangle$ *and* $\omega' = \langle \mathsf{X}'_{\tau_1':\tau_2'}\varphi' \wedge \bigwedge_{j=0}^{m}\gamma'_j, t'_d, t'_a, t'_e\rangle$ *(*X *and* X' *being either* O, F *or* P*) holds iff* Ω *is a possibly empty and finite set of norms obtained by curtailing* ω *with respect to* ω'*. The following cases arise:*

1. *If* **conflict**$(\omega, \omega', \sigma)$ *does not hold then* $\Omega = \{\omega\}$*, that is, the set of curtailments of a non-conflicting norm* ω *is* ω *itself.*
2. *If* **conflict**$(\omega, \omega', \sigma)$ *holds, then* $\Omega = \{\omega_0^c, \ldots, \omega_m^c\}$*, where* $\omega_j^c = \langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge (\neg\gamma'_j \cdot \sigma), t_d, t_a, t_e\rangle$*,* $0 \leq j \leq m$*.*

The rationale for the definition above is as follows. In order to curtail ω thus avoiding any overlapping of values its variables may have with those variables of ω', we must "merge" the negated constraints of ω' with those of ω. Additionally, in order to ensure the appropriate correspondence of variables between ω and ω' is captured, we must apply the substitution σ obtained via **conflict**$(\omega, \omega', \sigma)$ on the merged negated constraints. By combining the constraints of $\nu = \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i$ and $\nu' = \mathsf{X}'_{\tau_1':\tau_2'}\varphi' \wedge \bigwedge_{j=0}^{m}\gamma'_j$, we obtain the curtailed norm $\nu^c = \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\bigwedge_{j=0}^{m}\gamma'_j \cdot \sigma)$. The following equivalences hold:

$$\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\bigwedge_{j=0}^{m}\gamma'_j \cdot \sigma) \equiv \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge (\bigvee_{j=0}^{m}\neg\gamma'_j \cdot \sigma)$$

That is, $\bigvee_{j=0}^{m}(\mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg(\gamma'_j \cdot \sigma))$. This shows that each constraint of ν' leads to a possible solution for the resolution of a conflict and a possible curtailment of ν. The curtailment thus produces a set of curtailed norms $\nu_j^c = \mathsf{X}_{\tau_1:\tau_2}p(t_1, \ldots, t_n) \wedge \bigwedge_{i=0}^{n}\gamma_i \wedge \neg\gamma'_j \cdot \sigma, 0 \leq j \leq m$. Although each of the ν_j^c, $0 \leq j \leq m$, represents a solution to the norm conflict, we advocate that *all* of them have to be added to Ω in order to replace the curtailed norm. This would allow a preservation of as much of the original scope of the curtailed norm as possible. During the formation of a conflict-free Ω', the agent has to choose which norm to curtail in case of a conflict. In order to express such a choice, we introduce the concept of special curtailment *policies* that determine, given a pair of norms, which norm to curtail. We define curtailment policies as:

Definition 11. *A policy* π *is a tuple* $\langle \omega, \omega', (\bigwedge_{i=0}^{n}\gamma_i)\rangle$ *establishing that* ω *should be curtailed (and* ω' *should be preserved), if* $(\bigwedge_{i=0}^{n}\gamma_i)$ *hold.*

For example, a policy $\langle\langle \mathsf{F}_{A:R}p(X, Y), T_d, T_a, T_e\rangle, \langle \mathsf{P}_{A:R}p(X, Y), T'_d, T'_a, T'_e\rangle, (T_d < T'_d)\rangle$ expresses that any prohibition held by any agent that corresponds to the pattern $\mathsf{F}_{A:R}p(X, Y)$ has to be curtailed, if the additional constraint, which expresses that the prohibition's time of declaration T_d precedes that of the permission's T'_d, holds. Adding constraints to policies allows us a fine-grained control of conflict resolution, capturing classic forms of resolving deontic conflicts – the constraint in the example establishes a precedence relationship between the two norms that is known as *legis posterior* (see section 8 for more details). We shall represent a set of such policies as Π.

The algorithm shown in figure 4 depicts how to obtain a conflict-free set of norms. It describes how an originally conflict-free (possibly empty) set Ω can be extended in a fashion that resolves any emerging conflicts during norm adoption. With that, a conflict-free Ω is always transformed into a conflict-free

```
algorithm adoptNorm(ω, Ω, Π, Ω')
input ω, Ω, Π
output Ω'
begin
  Ω' := ∅
  if Ω = ∅ then Ω' := Ω ∪ {ω}
  else
    for each ω' ∈ Ω do
      // test for conflict
      if unify(ω, ω', σ) then
        // test policy
        if ⟨ω_π, ω'_π, (⋀_{i=0}^{n} γ_i)⟩ ∈ Π and unify(ω, ω_π, σ) and unify(ω', ω'_π, σ) and
          satisfy(⋀_{i=0}^{n}(γ_i · σ))
        then
          curtail(ω, ω', Ω'')
            Ω' := Ω ∪ Ω''
        else
          // test policy
          if ⟨ω'_π, ω_π, (⋀_{i=0}^{n} γ_i)⟩ ∈ Π and unify(ω, ω_π, σ) and unify(ω', ω'_π, σ) and
            satisfy(⋀_{i=0}^{n}(γ_i · σ))
          then
            curtail(ω', ω, Ω'')
            Ω' := (Ω − {ω'}) ∪ ({ω} ∪ Ω'')
          endif
        endif
      endif
    endfor
  endif
end
```

Fig. 4: Norm Adoption Algorithm

Ω' that may contain curtailments. The algorithm makes use of a set Π of policies determining how the curtailment of conflicting norms should be done. Policies determine whether the new norm ω is curtailed in case of a conflict or whether a curtailment of one of the existing $\omega' \in \Omega$ should take place. When a norm is curtailed, a set of new norms replace the original norm. This set of norms is collected into Ω'' by $\mathbf{curtail}(\omega, \omega', \Omega'')$. A curtailment takes place if there is a conflict between ω and ω'. This test creates a unifier σ that is re-used in the policy test. When checking for a policy that is applicable, the algorithm uses unification to check *(a)* whether ω matches/unifies with ω_π and ω' with ω'_π; and *(b)* whether the policy constraints hold under the given σ. If a previously agreed policy in Π determines that the newly adopted norm ω is to be curtailed in case of a conflict with an existing $\omega' \in \Omega$, then the new set Ω' is created by adding Ω'' (the curtailed norms) to Ω. If the policy determines a curtailment of an existing $\omega' \in \Omega$ when a conflict arises with the new norm ω, then a new set Ω' is formed by *a)* removing ω' from Ω and *b)* adding ω and the set Ω''.

5 Norm-Aware Agent Societies

With a set Ω that reflects a conflict-free normative situation, the agent can test whether its actions are norm-compliant. In order to check actions for norm-compliance, we, again, use unification. If an action unifies with a norm, then it is within its scope of influence:

Definition 12. *$\langle a : r, \bar{\varphi}, t\rangle$, is within the scope of influence of $\langle \mathsf{X}_{\tau_1:\tau_2}\varphi \wedge \bigwedge_{i=0}^{n}\gamma_i, t_d, t_a, t_e\rangle$ (where X is either O, P or F) iff the following conditions hold:*

1. *$unify(a, \tau_1, \sigma)$, $unify(r, \tau_2, \sigma)$, $unify(\bar{\varphi}, \varphi, \sigma)$ and $satisfy(\bigwedge_{i=0}^{n}\gamma_i \cdot \sigma)$*
2. *$t_a \leq t \leq t_e$*

This definition can be used to establish a 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 prohibition ω, its second parameter. Figure 5 shows

$$
\begin{array}{l}
\texttt{check}(Action, \omega) \leftarrow \\
\quad Action = \langle a : r, \bar{\varphi}, t\rangle \wedge \\
\quad \omega = \langle (\mathsf{F}_{\tau_1:\tau_2}\varphi' \wedge \bigwedge_{i=0}^{n}\gamma_i), t_d, t_a, t_e\rangle \wedge \\
\quad unify(\langle a, r, \bar{\varphi}\rangle, \langle \tau_1, \tau_2, \varphi'\rangle, \sigma) \wedge satisfy(\bigwedge_{i=0}^{n}\gamma_i \cdot \sigma) \wedge \\
\quad t_a \leq t \leq t_e
\end{array}
$$

Fig. 5: Check if Action is within Influence of a Prohibition

the definition of this relationship as a logic program. Similarly to the check of conflicts between norms, it tests *i)* if the agent performing the action and its role unify with the appropriate terms τ_1, τ_2 of ω; *ii)* if the actions $\bar{\varphi}, \varphi$ themselves unify; and *iii)* the conjunction of the constraints of both norms can be satisfied, all under the same unifier σ. Lastly, it checks if the time of the action is within the norm temporal influence.

6 Indirect Conflicts

In our previous discussion, norm conflicts were detected via a direct comparison of atomic formulae representing actions. However, conflicts and inconsistencies may also arise *indirectly* via relationships among actions. For instance, if we consider that an agent holds the two norms $\mathsf{P}_{A:R}p(X)$ and $\mathsf{F}_{A:R}q(X, X)$ and that the action $p(X)$ amounts to the action $q(X, X)$, then we can rewrite the permission as $\mathsf{P}_{A:R}q(X, X)$ and identify an *indirect* conflict between these two norms. We use a set of *domain axioms* in order to declare such domain-specific relationships between actions:

Definition 13. *The set of 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.*

In order to accommodate indirect conflicts between norms based on domain-specific relationships of actions, we have to adapt our curtailment mechanism. A curtailment occurs, if there is a conflict, that is, if for two norms ω and ω', their variables unify, the conjunction of their constraints can be satisfied and their activation periods overlap. With the introduction of domain axioms, this test has to be performed for each of the conjuncts in the relationship. For example, if we have a set of domain axioms $\Delta = \{(p(X) \rightarrow q(X, X) \wedge r(X, Y))\}$

and a permission $\langle \mathsf{P}_{A:R} p(X), t_d, t_a, t_e \rangle$ then $q(X, X)$ and $r(X, Y)$ are also permitted. There is, thus, an indirect conflict between $\langle \mathsf{P}_{A:R} p(X), t_d, t_a, t_e \rangle$ and $\langle \mathsf{F}_{A:R} q(X, X), t_d, t_a, t_e \rangle$ and $\langle \mathsf{F}_{A:R} r(X, Y), t_d, t_a, t_e \rangle$.

Domain axioms may also accommodate the delegation of actions between agents. Such a delegation transfers norms across the agent community and, with that, also conflicts. 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.

7 Example: Agents for the Grid

We address a scenario taken from the e-Science/Grid domain in which a service provider may request payment that introduces a financial obligation for users, but, at the same time commits to the provision of the service that represents a right for the user to access the service.

In this scenario, a Principal Investigator (PI) of a research project has to perform a specific research task that involves the analysis of data. We assume that a contract exists between the PI and the funding body that introduces certain rights, restrictions and obligations for the contracting partners. We regard both the PI and the funding body as being represented as agents operating on the Grid and that this contract is available in electronic form and taken into account by the agents in their actions.

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

Fig. 6: Contract C

A possible initial contract C is shown in Fig. 6. The first three norms represent 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. Contract C in its alternative (equivalent) format in which constants are replaced by variables and constraints is shown in Fig. 7.

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

Fig. 7: Alternative Format of Contract C

7.1 Conflict Resolution

Contract C has conflicting norms. We use our machinery to obtain a conflict-free version C' of it, in which only the first prohibition is curtailed. C' is shown in Fig. 8. In our example, two Grid services are made available by two potential subcontractors for the execution of the data analysis task. These are: *i)* a public non-profit organisation provides a free service, but

$$\left\{ \begin{array}{l} \langle \mathsf{F}_{A:R}\ claim(X) \wedge \left(\begin{smallmatrix} A = rsa \wedge R = pi \wedge \\ X \neq staff\_costs \wedge \\ X \neq travel \end{smallmatrix} \right), 1, 1, 1000 \rangle \\ \langle \mathsf{P}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000 \rangle \\ \vdots \\ \langle \mathsf{F}_{X:Y}\ publish(D), 1, 1, 1000 \rangle \end{array} \right\}$$

Fig. 8: Contract C' with Curtailed Norm

requires the disclosure of data in a public repository; and *ii)* a private commercial organisation 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 organisation, 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 organisation, on the other hand, are $\langle \mathsf{O}_{A:R}\ pay(fee), 1, 1, 1000\rangle$ or, alternatively, $N_2 = \{\langle \mathsf{O}_{A:R}\ pay(X) \wedge X = fee, 1, 1, 1000\rangle\}$ That is, whoever uses the service is obliged to pay a fee. The Research Assistant Agent (*rsa*) has to choose which service to use. Each of them introduces a new obligation with associated inconsistencies, explained below.

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 a conflict: the obligation to publish overlaps with the influence of the prohibition to publish. Our machinery handles this, completely curtailing the prohibition and giving rise to a new set C'', shown in Fig. 9. The constraint $D \neq D'$ expresses that variable D cannot be bound to anything (since D' is a free variable) – the prohibition, therefore, becomes completely curtailed and has no effect any more and, hence, it is removed.

$$\left\{\begin{array}{l}\langle \mathsf{F}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{P}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{P}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{O}_{A:R}\ report\_experiment(A, D) \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{F}_{X:Y}\ publish(D) \wedge D \neq D', 1, 1, 1000\rangle\\ \langle \mathsf{O}_{A:R}\ publish(D'), 1, 1, 1000\rangle\end{array}\right\}$$

Fig. 9: Contract $C'' = C' \cup N_1$

A conflict within the set $C' \cup N_2$ is not immediately obvious. Intuitively, in terms of paying expenses for research (the domain of discussion here), the action *pay* is related to the action *claim*. In order for our mechanism to cope with such a situation, a concept of *indirect* conflicts based on domain axioms for relating actions has to be introduced. We have explored such indirect conflicts in [2] and we plan to extend that work to handle arbitrary constraints.

7.2 Indirect Conflict Resolution

In choosing the private service, the obligation $N_2 = \{\langle \mathsf{O}_{A:R}\ pay(X) \wedge X = fee, 1, 1, 1000\rangle\}$ is introduced and a contract $C'' = C' \cup N_2$ created. Intuitively, we know that this introduces an *indirect* conflict, as the original contract does not allow such a claim. With a domain axiom, we can express that to pay for something eventually amounts to claiming it: $\Delta = \{pay(X) \xrightarrow{A:R\ A:R} claim(X)\}$. In contract C'', we have to permissions that allow claiming staff costs and travel, but not claiming fees. According to the given domain axiom, obligation N_2 can be transformed into $N_2^{\Delta} = \mathsf{O}_{A:R}\ claim(X) \wedge X = fee, 1, 1, 1000\rangle\}$. By forming a new contract $C'' = C' \cup N_2^{\Delta}$, a direct conflict between the first prohibition regarding claims and obligation N_2^{Δ} arises (Fig. 10). The conflict resolution can now take place as shown in the case of *direct* conflicts (see contract C' in Fig. 8).

$$\left\{\begin{array}{l}\langle \mathsf{F}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{P}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{P}_{A:R}\ claim(X) \wedge \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{O}_{A:R}\ report\_experiment(A, D) \ldots, 1, 1, 1000\rangle\\ \langle \mathsf{F}_{X:Y}\ publish(D) \wedge D \neq D', 1, 1, 1000\rangle\\ \langle \mathsf{O}_{A:R}\ claim(X) \wedge X = fee, 1, 1, 1000\rangle\end{array}\right\}$$

Fig. 10: Contract $C'' = C' \cup N_2^{\Delta}$

7.3 Solving Conflicts arising from Delegation

Conflicts can also arise from delegation among agents/roles. Let there be the set of domain axioms Δ of Fig. 11: it contains axioms describing 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. 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.

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

Fig. 11: Set of Domain Axioms Δ

8 Related Work

The work presented in this paper is an extension and adaptation of the work presented in [2, 16] and [17]. It can also be seen as a logic-theoretic investigation into deontic logics to represent normative modalities along with their paradoxes [18, 19]. In [2], we introduced conflict detection and resolution based on unification. In this paper, we re-visited this research and introduced constraints into the given conflict detection/resolution mechanism. The result is a new machinery for conflict detection/resolution and reported in this paper.

Efforts to keep law systems conflict-free can be traced back to the jurisprudential practice in human society. 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 by establishing a precedence relationship between norms: *legis posterior* – the most recent norm takes precedence, *legis superior* – the norm imposed by the strongest power takes precedence, and *legis specialis* – the most specific norm takes precedence [20]. The work presented in [16] discusses a set of conflict scenarios and conflict resolution strategies, among them the classic strategies mentioned above. For example, one of these conflict resolution strategies achieves a resolution of a conflict via negotiation with a norm issuer. In [21], an analysis of different normative conflicts is provided. The authors suggest that a deontic inconsistency arises when an action is simultaneously permitted and prohibited. In [22], three forms of conflict/inconsistency are described as *total-total*, *total-partial* and *intersection*. These are special cases of the intersection of norms as described in figure 2 and in [16] – a permission entailing the prohibition, a prohibition entailing the permission or an overlap of both norms.

The *S*CIFF framework [23] is related to our work in that it also uses constraint resolution to reduce the scope of expectations to avoid conflict – expectation is a concept closely related to norms [24]. For instance, in that work, $\mathbf{E}(p, X), 0 \leq X \leq 10$ means that p is expected to hold true between 0 and

10, and $\mathbf{EN}(p, Y), Y > 5$ means that p is expected not to hold true when Y is greater than 5; positive expectations are related to obligations (and are implicitly existentially quantified) and negative expectations are related to prohibitions (and are implicitly universally quantified). The *S*CIFF proof procedure uses constraint resolution to reduce the domain of the expectations (and non-expectations). However, *S*CIFF always gives higher priority to negative expectations against positive ones.

9 Conclusions and Future Work

We have presented a novel mechanism to detect and resolve conflicts in norm-regulated environment. Such conflicts arise when an action is simultaneously obliged and prohibited or, alternatively, when an action is permitted and prohibited. We introduce norms as first-order atomic formulae to whose variables we can associate arbitrary constraints – this allows for more expressive norms, with a finer granularity and greater precision. The proposed mechanism is based on first-order unification and constraint satisfaction algorithms, extending our previous work [2], addressing a more expressive class of norms. Our conflict resolution mechanism amounts to manipulating the constraints of norms to avoid overlapping values of variables – this is called the "curtailment" of variables/norms. We have also introduced a robust and flexible algorithm to manage the adoption of possibly conflicting norms, whereby explicit policies depict how the curtailment between specific norms should take place. Our proposed formalism naturally allows the detection of indirect normative conflicts, arising when an action is broken down into composite actions appearing in conflicting norms.

In this paper we only considered universally quantified norms, leaving out important cases of existential quantifications. If existential quantification is allowed, then disjunction of constraints must be preserved. In this case, replacing a norm that has a disjunction of constraints with a conjunction of separate norms does not work anymore. If we allow existential quantification then we must preserve disjunctions of constraints and the set of norms Ω should be managed differently, in particular, disjunctions of norms should be allowed. We are currently working to address these issues.

The policies establishing which of two conflicting norms should be curtailed, confers generality on our approach, being neatly accommodated in our algorithms. We observe, however, that it would also be possible to make policies part of the virtual organisation (VO) specification, giving higher priority to those norms that allow the progress of the organisation. For instance, if $p(X)$ is forbidden and $p(Y)$ is permitted (both for the same group of agents/roles), that is, there is a complete overlap on the norms' scope of influence, then a policy on the VO could specify which of the two should be "removed" (by adding the constraint $X \neq Y$ onto it), based on which of them would allow the VO to progress. For example, if the VO progresses when an agent performs $p(a)$, then the prohibition could be lifted.

We want to extend our work to also address the removal of norms: when a norm is removed, all those curtailments it caused must be undone. We envisage a roll-back/roll-forward mechanism, whereby a history of normative states allows us to retrieve the state prior to the introduction of the norm to be removed (roll-back) and apply to this state all the updates which took place after the norm was

introduced, skipping the actual norm to be removed (roll-forward). Additionally, we want to integrate our mechanisms with norm-updating approaches such as [12] – we want to investigate if it is possible (and in which circumstances) to detect conflicts at the design stage of norm updates (as opposed to run-time).

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 (`http://www.usukita.org`).

References

1. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer-Verlag, New York, U.S.A. (1990)
2. Vasconcelos, W., Kollingbaum, M., Norman, T., García-Camino, A.: Resolving Conflict and Inconsistency in Norm-Regulated Virtual Organizations. In: Proceedings of AAMAS 2007. (2007)
3. O'Leary, D.E., Kuokka, D., Plant, R.: Artificial Intelligence and Virtual Organizations. Commun. ACM **40**(1) (1997)
4. Apt, K.R.: From Logic Programming to Prolog. Prentice-Hall, U.K. (1997)
5. Parunak, H.V.D., Odell, J.: Representing Social Structures in UML. In: Procs 5th Int'l Conf. on Autonomous Agents, Montreal, Canada, ACM Press (2001) 100–101
6. Rodríguez-Aguilar, J.A.: On the Design and Construction of Agent-mediated Electronic Institutions. PhD thesis, IIIA-CSIC, Spain (2001)
7. Pacheco, O., Carmo, J.: A Role Based Model for the Normative Specification of Organized Collective Agency and Agents Interaction. Autonomous Agents and Multi-Agent Systems **6**(2) (2003) 145–184
8. Garcia-Camino, A., Rodriguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.W.: A Distributed Architecture for Norm-Aware Agent Societies. Volume 3904 of LNAI. Springer-Verlag (2005)
9. Vasconcelos, W.W.: Expressive Global Protocols via Logic-Based Electronic Institutions. In: Proc. 2nd Int'l Joint Conf. on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, ACM, U.S.A (2003)
10. Pacheco, O., Carmo, J.: A Role Based Model for the Normative Specification of Organized Collective Agency and Agents Interaction. Autonomous Agents and Multi-Agent Systems **6**(2) (2003) 145–184
11. Jaffar, J., Maher, M.J.: Constraint Logic Programming: A Survey. Journal of Logic Progr. **19/20** (1994) 503–581
12. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: A Rule-based Approach to Norm-Oriented Programming of Electronic Institutions. ACM SIGecom Exchanges **5**(5) (2006) 33–40
13. Swedish Institute of Computer Science: SICStus Prolog. (2005) `http://www.sics.se/isl/sicstuswww/site/index.html`, viewed on 10 Feb 2005 at 18.16 GMT.
14. Jaffar, J., Maher, M.J., Marriott, K., Stuckey, P.J.: The Semantics of Constraint Logic Programs. Journal of Logic Programming **37**(1-3) (1998) 1–46
15. Holzbaur, C.: ÖFAI clp(q,r) Manual, Edition 1.3.3. TR-95-09, Austrian Research Institute for A. I., Vienna, Austria (1995)
16. Kollingbaum, M., Norman, T., Preece, A., Sleeman, D.: Norm Refinement: Informing the Re-negotiation of Contracts. In Boella, G., Boissier, O., Matson, E., Vazquez-Salceda, J., eds.: ECAI 2006 Workshop on Coordination, Organization, Institutions and Norms in Agent Systems, COIN@ECAI 2006. (2006) 46–51
17. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: An Algorithm for Conflict Resolution in Regulated Compound Activities. In: Seventh Annual International Workshop Engineering Societies in the Agents World (ESAW'06). (2006)

18. Dignum, F.: Autonomous Agents with Norms. Artificial Intelligence and Law **7** (1999) 69–79
19. Sergot, M.: A Computational Theory of Normative Positions. ACM Transactions on Computational Logic **2**(4) (2001) 581–622
20. Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-Dimensional Dynamic Knowledge Representation. Volume 2173 of LNAI. Springer-Verlag (2001)
21. Elhag, A., Breuker, J., Brouwer, P.: On the Formal Analysis of Normative Conflicts. Information & Comms. Techn. Law **9**(3) (2000) 207–217
22. Ross, A.: On Law and Justice. Stevens & Sons (1958)
23. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: The SCIFF Abductive Proof Procedure. Volume 3673 of LNAI. Springer-Verlag (2005)
24. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Sartor, G., Torroni, P.: Mapping Deontic Operators to Abductive Expectations. Computational & Mathematical Organization **12**(2-3) (2006) 205–225