Using constraints for Norm-aware BDI Agents

Felipe Meneguzzi
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
meneguzz@cs.cmu.edu

Wamberto Vasconcelos
Dept. of Computing Science
University of Aberdeen
Aberdeen, UK
wvasconcelos@acm.org

Nir Oren
Dept. of Computing Science
University of Aberdeen
Aberdeen, UK
n.oren@abdn.ac.uk

***Abstract*—Systems of autonomous and self-interested agents interacting to achieve individual and collective goals may exhibit undesirable or unexpected properties if left unconstrained. Using deontic concepts of obligations, permissions and prohibitions to describe, what must, may and should not be done, norms have been widely proposed as a means of defining and enforcing societal constraints. Recent efforts to provide norm-enabled agent architectures that limit plan choices suffer from interfering with an agent's reasoning process, and thus limit autonomy more than is required by the norms alone. In response, in this paper we describe nu-BDI, an extension of the BDI architecture, which enables normative reasoning, providing agents with a means to choose and customise plans (and their constituent actions), so as to ensure compliance with norms. We make three significant contributions, in providing: fine-grained tailoring of plan restrictions; a plan annotation mechanism to identify violating plans, and limit possible plan instantiations; and a technique allowing the selective and incremental violation of norms in cases where goal achievement would not otherwise be possible.**

I. Introduction

Systems of autonomous and self-interested agents interacting to achieve individual and collective goals may exhibit undesirable or unexpected properties if left unconstrained. One way to address this issue, in both human and artificial societies, has been through the use of norms, which have been proposed as a means of defining and enforcing constraints to ensure that such undesired behaviour is avoided if agents are norm-compliant (cf. [11] and [6]). Norms are generally specified using deontic concepts of obligations, permissions and prohibitions to identify, respectively, what must, may and should not be done so as to ensure certain system properties. Early work on normative systems focused on model-theoretic or philosophical aspects of deontic logics [18], but more recent work has addressed how norms may be more suitably represented in computational systems (*e.g.*, [13] and [14]), their enforcement [8], and their impact on the society as a whole, abstracting away the details of mechanisms through which individual agents reason with and about norms and how individual behaviours are affected by norms.

However, practical normative systems require analysis and specification of the processes through which norms are recognised, decisions about whether to comply with them are taken, and behaviour is adjusted appropriately. Some recent efforts in this direction have sought to provide norm-enabled architectures (*e.g.*, [12] and [15]) to specify how an agent's behaviour may be constrained to comply with norms in terms of permitted or forbidden mental states. For example, compliance with an obligation to move to a certain location limits an agent's choice of plans containing moving actions to only those in which the target of the actions is the obliged location. While such architectures capture this notion at a basic level, for example in preventing parts of a plan library from being adopted [15], or replacing the goals of an agent with the fulfilment of specific norms [12], they suffer from interfering with an agent's reasoning process, and thus limit autonomy more than is required by the norms alone.

In response, in this paper we introduce ν-BDI, an extension of the BDI architecture [16] that enables normative reasoning, and provides a means for agents to choose and customise plans (and their constituent actions), so as to ensure compliance with norms. The paper makes three significant contributions. First, it avoids the rather coarse blanket retraction of specific plans (as adopted in previous work) by introducing *constraints*, enabling fine-grained tailoring of plan restrictions. Second, it provides a plan annotation mechanism used as an efficient means of identifying (and potentially avoiding) plans that violate norms by examining norm scope (in relation to actions), and limiting possible plan instantiations. This effectively transforms normative restrictions into extended context conditions that incur a similar computational overhead as selecting a plan. Finally, it provides a technique for the selective and incremental violation of norms in cases where goal achievement would not be possible otherwise. Importantly, unlike some earlier efforts, such agents are able to comply with specific normative stipulations with minimum disruption to traditional non-norm influenced reasoning.

To illustrate our approach, we adopt a scenario in which software agents support humans responding to an emergency situation. Humans communicate with each other and synchronise their activities through personal assistants responsible for intermediating communication among members of a team, prompting their human counterparts for actions to be carried out, as well as providing information to help humans decide which course of action to take. We address the situation in which heavy and continuous rain in an area prone to flooding has led to emergency services being put on alert: a team of humans supported by personal assistants is to carry out alternative plans, depending on the current conditions (e.g., severity of the flooding, size of the affected area, which buildings are more at risk, the people affected, etc.). The personal

assistants monitor the latest information on weather and rising levels of water, and also have access to data on high-security installations (e.g., power plants, fuel and chemical depots, etc.), high-risk buildings (e.g., hospitals with intensive care patients, primary schools, prisons, etc.), routes for evacuation, and so on. They have the following plan available, which we use to illustrate our approach:

> If one detects that the level of flooding in an area X is medium, and if the area is of high-risk (that is, it contains high-risk buildings), then the plan is to: *i)* isolate the area (to prevent people entering it); *ii)* evacuate everyone from the affected area to another area; and *iii)* reroute the traffic to another area.

In our scenario we also assume two norms:

1) It is forbidden to evacuate an area X to an area Y, if area Y is unsafe. This prohibition should be revoked if area Y becomes safe.
2) It is obligatory to reroute traffic through Z to avoid area X, if area X is deemed not safe. This obligation is revoked when area X becomes safe again.

We start the paper by reviewing the BDI agent model and introducing a basic interpreter in Section II. In order to define our proposed extension, we introduce in Section III a notation for precisely specifying normative restrictions, including restrictions over acceptable domains. Using this notation, we develop in Section IV an agent architecture capable of reasoning with these norms, thus affecting specific plan instances that are adopted, deciding on norm compliance as plan instances are selected. In doing so, we fulfil the need of pragmatic normative agent architectures capable of filtering norm compliant plans and decide upon them. Finally, we draw conclusions and point to future work in Section V.

II. BDI Reasoning

In this section we review the well-known BDI architecture, based on Bratman's philosophical model of reasoning centred around the three mental components of beliefs, desires and intentions (BDI) [2]. We use this as the foundation of our norm-aware architecture.

A. Preliminaries

In order to explain the operation of ν-BDI, we need to introduce some notation and definitions. We use first-order constructs for various elements of the agents and norms.

***Definition** 1:* A term, denoted generically as τ, is a variable w, x, y, z (with or without subscripts), a constant a, b, c (with or without subscripts) or $f^n(\tau_0, \ldots, \tau_n)$, that is, an n-ary function f^n applied to (possibly nested) terms $\tau_0, \ldots, \tau_n$.

***Definition** 2:* A first-order atomic formula (or a predicate), denoted as φ, is any construct of the form $p^n(\tau_0, \ldots, \tau_n)$, where p^n is an n-ary predicate symbol applied to terms $\tau_0, \ldots, \tau_n$. A first-order formula, denoted as Φ, is defined as $\Phi ::= \Phi \wedge \Phi | \neg\Phi | \forall x.\Phi | \varphi$.

We assume the usual abbreviations: $\Phi \vee \Phi'$ stands for $\neg(\neg\Phi \wedge \neg\Phi')$, $\exists x.\Phi$ stands for $\neg\forall x.\neg\Phi$, $\Phi \rightarrow \Phi'$ stands for $\neg\Phi \vee \Phi'$ and $\Phi \leftrightarrow \Phi'$ stands for $(\Phi \rightarrow \Phi') \wedge (\Phi' \rightarrow \Phi)$. Additionally, we also adopt the equivalence $\{\Phi_1, \ldots, \Phi_n\} \equiv (\Phi_1 \wedge \cdots \wedge \Phi_n)$ and use these interchangeably. In our mechanisms we use first-order unification [5] which is based on the concept of substitutions.

***Definition** 3:* 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 follows:

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

Unifications can be *composed*; that is, for any $\sigma_1 = \{x_1/\tau_1, \ldots, x_n/\tau_n\}$ and $\sigma_2 = \{y_1/\tau'_1, \ldots, y_k/\tau'_k\}$, their composition, denoted as $\sigma_1\sigma_2$, is defined as $\{x_1/(\tau_1 \cdot \sigma_2), \ldots, x_n/(\tau_n \cdot \sigma_2), z_1/(z_1 \cdot \sigma_2), \ldots, z_m/(z_m \cdot \sigma_2)\}$, where $\{z_1, \ldots, z_m\}$ are those variables in $\{y_1, \ldots, y_k\}$ that are not in $\{x_1, \ldots, x_n\}$. A substitution σ is a *unifier* of two terms τ_1, τ_2, if $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$.

***Definition** 4:* $unify(\tau_1, \tau_2, \sigma)$ holds iff $\tau_1 \cdot \sigma = \tau_2 \cdot \sigma$, for some σ. $unify(p^n(\tau_0, \ldots, \tau_n), p^n(\tau'_0, \ldots, \tau'_n), \sigma)$ holds iff $unify(\tau_i, \tau'_i, \sigma), 0 \leq i \leq n$.

Two terms τ_1, τ_2 are related through the $unify$ relation if there exists a substitution σ that makes the terms syntactically equivalent. We assume a suitable implementation of a unification algorithm for $unify$ to determine the existence of such a substitution.

We denote as $\bar{\varphi}$ a first-order predicate whose terms are either constants or variables associated (via a substitution) with constants. Here, 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. We assume the availability of a sound and complete first-order inference mechanism[1] which decides if Φ' can be inferred from Φ, denoted as $\Phi \vdash \Phi'$. In this paper we use a mechanism to determine if a formula Φ can be inferred from a set of ground predicates, and, if so, under which substitution; that is, $\{\bar{\varphi}_0, \ldots, \bar{\varphi}_n\} \vdash \Phi \cdot \sigma$.

B. Agents

In this work we consider an abstract BDI interpreter, inspired by the dMARS architecture [4]. We define an agent in terms of its information model as follows.[2]

***Definition** 5:* An agent is a tuple $\langle Ag, Rl, Ev, Bel, Plib, Int \rangle$, where Ag is the agent identifier, Rl is a set of roles, Ev is a queue of events, Bel is a belief base, $Plib$ is a plan library, and Int is an intention structure.

The Rl component is a finite and non-empty set of roles $\{r_1, \ldots, r_n\}$, used to identify stereotypical agents classes to which one belongs (*e.g.* $\{fire\_marshall, evacuation\_team\}$). Recently perceived events are stored in a queue and ordered

[1] Such mechanisms have a design space defined by the expressiveness of the language and complexity/decidability aspects – the more expressive the language, the fewer guarantees can be given [5]. In particular, if we assume our first-order language is restricted to Horn clauses, then we can use Prolog's resolution mechanism [1].

[2] We use Greek letters to denote elements of the underlying logic system as well as norms, and use Latin letters to denote elements of the agent interpreter.

by arrival time. An event may be a belief addition or deletion, or a goal addition or deletion. Belief additions are *positive ground predicates* perceived as true, and belief deletions are *negative ground predicates* perceived as false. Goal additions indicate new goals posted, and goal deletions represent goals dropped for some reason.

***Definition** 6:* An event queue Ev is composed of ground first-order predicates representing events $[e_1, \ldots, e_n]$ ordered by occurrence time. Events e_i can be one of four possible cases: *i)* a belief addition $+\bar{\varphi}$; *ii)* a belief deletion $-\bar{\varphi}$; *iii)* a goal addition $+!\bar{\varphi}$; or *iv)* a goal deletion $-!\bar{\varphi}$.

The belief base comprises a set of logic predicates, which can be queried through an entailment relation, as follows.

***Definition** 7:* A belief base Bel is a finite and possibly empty set of ground first-order logic predicates $\{\bar{\varphi}_1, \ldots, \bar{\varphi}_n\}$, with an associated logical entailment relation $\vdash$ for first-order formulae.

The plan library, defined below, stores the plans of action available. Each step in a plan body may be either an action (causing effects in the environment) or a subgoal (causing the addition of a new plan to the intention structure).

***Definition** 8:* A plan library $Plib$ is a finite and possibly empty set of uninstantiated plans $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. Each plan $\mathcal{P}_i$ is a tuple $\langle t, c, bd \rangle$ where t is an invocation condition (cf. Definition 6), indicating the event that causes the plan is to be adopted, c is a context condition in the form of a first-order formula over the agent's belief base, and bd is a body consisting of a finite and possibly empty sequence of steps $[s_0, \ldots, s_n]$.

Actions are first-order atomic formulae; our focus is not on what an action entails, just that action execution might be the target of a normative stipulation as we will see later. Finally, the intention structure comprises the agent's intentions, each of which contains partially instantiated plans to be executed by the agent.

***Definition** 9:* An intention structure Int is a finite and possibly empty set of intentions $\{int_1, \ldots, int_n\}$. Each int_i is a tuple $\langle \sigma, \bar{st} \rangle$, where σ is a substitution and $\bar{st}$ is an intention stack (containing the steps remaining to be executed to achieve the intention).

EXAMPLE. The plan of our scenario is represented as follows:

$$\left\langle \; +level(X, medium), (high\_risk(X)), \left[\begin{array}{c} isolate(X), \\ evacuate(X, Y), \\ reroute(X, Z) \end{array} \right] \; \right\rangle$$

This represents that if a belief $level(X; medium)$ has been added to the belief base, stating that the level of emergency of area X is medium, and the context condition "X is a $high\_risk$ area" holds, then the plan should be adopted.

C. A BDI Agent Interpreter

The specification above provides a minimal information model required for BDI agent execution. In this section we describe the mechanisms needed (using this information) for BDI-style computational behaviour. Before considering norms, we specify a basic abstract BDI agent interpreter and subsequently extend it with mechanisms for normative reasoning and compliance. A high-level description of the basic BDI interpreter is illustrated as the white boxes in Figure 1; grey boxes are our proposed extensions.below. Initially, new events are perceived from the environment and added to an event queue Ev, then, the new events are used to update a belief base Bel and to select new plans from a plan library $Plib$ to be adopted as intentions in the intention structure Int. Finally, an intention is selected and one of its steps executed.

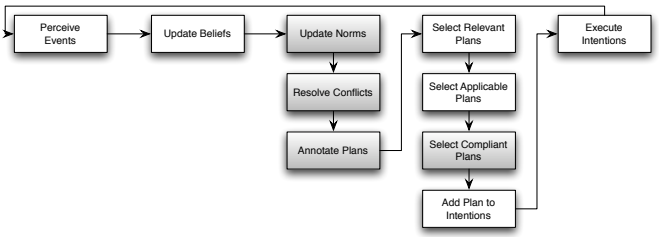

Fig. 1. Control flow for the reasoning process

Updating events consists of gathering all new events and pushing them onto the event queue, while updating beliefs consists of querying new events in the event queue and adding beliefs when the events are positive predicates, and removing them when they are negative predicates. We are not concerned here with more complex belief revision mechanisms, but such an interpreter could use them [7].

New events trigger the adoption plans from the plan library. If the event is a belief update, a new intention may be created for it, otherwise the event is a subgoal for some existing intention and the plan is added to it. The steps of a plan are adopted as intentions for execution, each of which can be either an action in the environment or the adoption of a subgoal that will trigger the adoption of further plans.

These procedures have a very low computational cost, as demonstrated by various practical implementations such as dMARS [4], PRS [9], and others. We now proceed to considering the normative aspects and how we incorporate them to the basic BDI interpreter.

III. REPRESENTATION OF NORMS

Using deontic concepts of obligations, permissions and prohibitions to describe, what must, may and should not be done, norms have been widely proposed as a means of defining and enforcing societal constraints. In this paper, since we are concerned with the impact of norms on reasoning and behaviour, we pay particular attention to the scope of influence of norms. In this respect, we consider two distinct means of addressing this: first, we draw on aspects similar to those presented in [6] and [12] in that our norms are *conditional*, both for activation (when they come into force) and expiration (when they cease effect), limiting application to periods of time; and second, we add *constraints* [10], limiting application to particular plans and actions, and ensuring that norms are not over-restrictive. In this section, we adapt and extend the notation for specifying norms of [17], beginning with constraints.

***Definition** 10:* Constraints, represented as γ, are any construct of the form $\tau \lhd \tau'$, where τ, τ' are first-order terms (that is, a variable, a constant or a function applied to terms) and $\lhd$ is one of the infix binary operators $=, \neq, >, \geq, <$, or $\leq$. A conjunction of constraints is denoted as $\Gamma = (\gamma_1 \wedge \cdots \wedge \gamma_n)$.

We use numbers and arithmetic functions to build terms τ; arithmetic functions may appear infix, following their usual conventions. For example, $10 > \mathit{Temp}$ and $\mathit{Price} < (\mathit{Cost} + Z)$. To improve readability, constraints of the form $3 \leq X \wedge X \leq 10$ are written as $3 \leq X \leq 10$. Since constraints limit the acceptable range of parameters within instantiated plan steps (as we will see later), when determining if a plan complies with current norms, their satisfiability must be checked. We use existing constraint satisfaction techniques [10] to implement a *satisfy* predicate that holds if a given conjunction of constraints admits a solution (if each variable of the constraints admits at least one value that simultaneously fulfils all constraints).

***Definition** 11:* $\mathit{satisfy}(\gamma_0 \wedge \cdots \wedge \gamma_n, \sigma)$ holds iff $(\gamma_0 \cdot \sigma \wedge \cdots \wedge \gamma_n \cdot \sigma)$ is true for some σ.

Constraints are associated with first-order predicates, imposing restrictions on their variables. We represent this association as $\varphi \circ \Gamma$, as in, for instance, $\mathit{move}(b_1, X, Y) \circ (100 \leq X \leq 500 \wedge 5 \leq Y \leq 45)$. Now, to define for the core aspect of norms, we use constraint-annotated atomic deontic formulae.

***Definition** 12:* An annotated deontic formula ν is any construct of the form $\mathsf{O}_{\alpha:\rho}\varphi \circ \Gamma$ (an obligation) or $\mathsf{F}_{\alpha:\rho}\varphi \circ \Gamma$ (a prohibition), where α, ρ are terms, and φ is a first-order atomic formula with associated constraints Γ.

Term α identifies the agent(s) to which the norm is applicable and ρ is the role of such agent(s). $\mathsf{O}_{\alpha:\rho}\varphi \circ (\gamma_1 \wedge \ldots \wedge \gamma_n)$ thus represents an obligation on agent α taking up role ρ to bring about φ, subject to *all* constraints γ_i, $0 \leq i \leq n$. The γ_i terms express constraints on variables of φ. The relation between constraints and the deontic formula is akin to the *quantifier restrictions* introduced in [3]. If we assume a universal quantification in our annotated deontic formulae, that is, $\forall\alpha.\forall\rho.\forall\vec{x}(\mathsf{X}_{\alpha:\rho}\varphi \circ \Gamma)$ (where $\vec{x}$ are all variables occurring in φ and Γ, and X is either O or F) then our formula stands for $\forall\alpha.\forall\rho.\forall\vec{x}(\Gamma \rightarrow \mathsf{X}_{\alpha:\rho}\varphi)$. Alternatively, if we assume an existential quantification, that is, $\exists\alpha.\exists\rho.\exists\vec{x}(\mathsf{X}_{\alpha:\rho}\varphi \circ \Gamma)$, then the formula stands for $\exists\alpha.\exists\rho.\exists\vec{x}(\Gamma \wedge \mathsf{X}_{\alpha:\rho}\varphi)$.

Our representation here is precise (as constraints provide a fine-grained way to specify values of variables) and compact (as constrained predicates amount to possibly infinite sets of ground formulae). Let us assume the deontic formulae $\{\mathsf{F}p(X) \circ \{X = a\}, \mathsf{O}q(Y) \circ \{Y = b\}\}$ are currently in effect. We also assume that at a particular point there is the following choice of plans to achieve a particular goal (for brevity, we assume the agent and role are known, dropping the subscripts from the formulae, and simplifying the formulae as $\{\mathsf{F}p(a), \mathsf{O}q(b)\}$, respectively):

1) $[s(a,b), \boxed{p(a)}, q(a), r(a)]$
2) $[q(a), p(b), s(a,b), r(a)]$
3) $[\boxed{q(b)}, p(b), s(a,b), r(a)]$

A rational agent should give priority to Plan 3, which fulfils the obligation (shown boxed) and does not violate the prohibition. Plan 2 neither fulfils the obligation nor violates the prohibition. Plan 1 is the worst choice as it violates the prohibition (boxed). More interesting situations arise when plans both fulfil obligations and violate prohibitions. We develop in Section IV-C a means to manipulate plans, annotating them with constraints on the values of variables of its actions, thus ensuring that all norms in effect are factored in. In Section IV-D we propose a means for agents to rank plans according to their norm-compliance. Thus, norms are defined as follows.

***Definition** 13:* An abstract norm ω^A is a tuple $\langle \nu, \mathit{Act}, \mathit{Exp}, \mathit{id} \rangle$ where:

- ν is an annotated deontic formula (cf. Def. 12),
- Act, the **activation condition**, is a conjunction of possibly negated first-order atomic formulae $\varphi_1 \wedge \cdots \wedge \varphi_n$ specifying the condition that must hold in the agent's belief base for the norm to take effect;
- Exp, the **expiration condition**, is a conjunction of possibly negated first-order atomic formulae $\varphi_1 \wedge \cdots \wedge \varphi_n$ specifying the condition that must hold in the agent's belief base for the norm to stop being in effect;
- id is a unique norm identifier

We denote a set of abstract norms as Ω^A. If the activation condition of an abstract norm holds, then a *specific norm* is obtained, whereby variables may be instantiated to specific values. Abstract norms generically define circumstances when norms should be adopted and dropped; when norms are adopted, the abstract formulation is instantiated to specific circumstances.

***Definition** 14:* A specific norm ω^S is a tuple $\langle \nu, \mathit{Act}, \mathit{Exp}, \sigma, \mathit{id} \rangle$ where $\nu, \mathit{Act}, \mathit{Exp}, \mathit{id}$ are as above and are bound by a substitution σ. We denote a set of specific norms as Ω^S.

As agents interact with their environment and with other agents, their perception of reality, as recorded in their sets of beliefs, change. Agents use their beliefs to update their normative positions, adding norms whose activation conditions hold, and removing norms whose expiration conditions holds. Given a set of beliefs Bel and a specific norm ω^S of the form $\langle \nu, \mathit{Act}, \mathit{Exp}, \sigma, \mathit{id} \rangle$, then ω^S holds (or is in effect) if, and only if, the following two conditions hold:

1) $\mathit{Bel} \vdash \mathit{Act} \cdot \sigma$; that is, we can deduce $\mathit{Act} \cdot \sigma$ from the set of beliefs, and
2) $\mathit{Bel} \nvdash \mathit{Exp} \cdot \sigma$; that is, we cannot deduce $\mathit{Exp} \cdot \sigma$ from the set of beliefs.

Since beliefs change, norms also change as their activation and expiration conditions may no longer hold; this is how dynamic aspects are captured in our representation of norms.

EXAMPLE. The norms of our scenario are represented as the following abstract norms:

1. $\langle \mathsf{F}_{A:R}\mathit{evacuate}(X,Y) \circ \{Y = W\}, \neg\mathit{safe}(W), \mathit{safe}(W), 1 \rangle$
2. $\langle \mathsf{O}_{A:R}\mathit{reroute}(X,Z) \circ \{X+1 \leq Z \leq X+3\}, \neg\mathit{safe}(X), \mathit{safe}(X), 2 \rangle$

The first norm states that all agents (in all roles) are forbidden to evacuate an area X to an area Y; the prohibition becomes active if area Y (constrained to be W) is unsafe and expires

when area Y (constrained to be W) becomes safe; unifications are dealt with like constraints, hence the need to use a third variable W. The second norm states that all agents (in all roles) are obliged to reroute traffic through Z to avoid area X, but the rerouting must be within nearby zones. The norm becomes active when area X is deemed not safe, and the norm is deactivated when area X becomes safe again. Now, suppose these norms give rise to the following specific norms:

3. $\langle \mathsf{F}_{A:R} evacuate(X,Y) \circ \{Y = W\}, \neg safe(W), safe(W), \{W/3\}, 1\rangle$
4. $\langle \mathsf{F}_{A:R} evacuate(X,Y) \circ \{Y = W\}, \neg safe(W), safe(W), \{W/6\}, 1\rangle$
5. $\langle \mathsf{O}_{A:R} reroute(X,Z) \circ \Gamma, \neg safe(X), safe(X), \{X/2\}, 2\rangle$

That is, abstract Norm 1 gives rise to two specific norms, one instantiating W to 3 and another W to 6. Abstract Norm 2 (shown with constraints abbreviated as Γ to save space) gives rise to one specific norm, instantiating X to 2.

For simplicity, in our discussion we assume an implicit universal quantification over variables in ν, Act and Exp. However, our approach can naturally be extended to cope with any quantification.

IV. ν-BDI

Given the representation of norms as detailed above, we can now address the issues surrounding their integration into an effective BDI architecture.

A. Updating Norms

First, we describe the key processes required in the agent interpreter to manage the activation and expiration of norms. Although beliefs are generally [4] assumed to contain exclusively ground first-order predicates, in this paper we store both abstract and specific norms in the belief base. In doing so we avoid adding extra components to the architecture. We extend and adapt the mechanisms to update beliefs and to reason with beliefs, enabling them to deal with norms.

The process of updating norms consists of going through each abstract norm $\omega^A \in \Omega^A$, of the form $\omega^A = \langle \nu, Act, Exp, id\rangle$, checking if their activation condition is supported by the agent's belief base, that is, $Bel \vdash Act \cdot \sigma$. Then, for each norm and each possible substitution σ in which the activation condition holds in Bel, a new specific norm $\omega^S = \omega^A \cdot \sigma$ is created and added to the set of specific norms. Afterwards, for each specific norm $\omega^S \in \Omega^S$, if the expiration condition is supported by the agent's belief base, that is, $Bel \vdash Exp \cdot \sigma\sigma'$, the specific norm is removed from Ω^S.

EXAMPLE. Let us suppose we have an abstract norm ω^A:

$$\langle \mathsf{O}_{A:R} use(hlc, X) \circ \Gamma, high\_risk(X), weather(X, poor), 3\rangle$$

This represents an obligation on all agents/roles to fly a helicopter (represented as hlc) over X; the norm becomes active if X is a high-risk area, and the norm expires if the weather conditions in X are poor. The Γ stipulates which areas can be flown over, and its details are not relevant to our example. If we have a belief base $Bel = \{high\_risk(10), \omega^A\}$, where ω^A is the abstract norm above, then we would add to Bel the specific norm

$$\langle \mathsf{O}_{A:R} use(hlc, X) \circ \Gamma, high\_risk(X), weather(X, poor), \{X/10\}, 3\rangle$$

If, however, the belief base also had a predicate $weather(10, poor)$, then no specific norms would be added, as the expiration condition of the newly added norm would hold – the mechanism would add and subsequently remove a specific norm, leaving the set of specific norms unchanged.

B. Actions and Norms

As indicated previously, our key concern in this paper is with the impact of norms on plans. Critical to this is determining when an action (represented as an atomic formula φ) is within the scope of influence of a specific norm ω^S. Definition 15 introduces predicate $inScope$ which, given an agent specified by its unique identifier Ag and one of the roles $R \in Rl$ of the agent, holds if a first-order predicate φ is within the influence of a specific norm ω^S (in the format of Definition 14).

***Definition** 15:* An action literal φ of an agent Ag with role R is in the scope of a specific norm $\omega^S = \langle \mathsf{X}_\alpha \varphi' \circ \Gamma, Act, Exp, \sigma, id\rangle$, represented as $inScope(Ag, R, \varphi, \omega^S)$, if, and only if, $\omega^S \in \Omega^S$, $unify(\langle Ag, R, \varphi\rangle, \langle \alpha, \rho, \varphi'\rangle \cdot \sigma, \sigma')$, and $satisfy(\Gamma \cdot \sigma, \sigma')$

C. Annotating Constraints in Plans

As indicated in Section III, one of our primary concerns is with the impact of norms on agent plans in terms of constraints on the values of variables of an action. Since actions and achievable world-states are components of plans, instances of restricted actions and world states must be found and marked with these constraints. To achieve this, we propose a mechanism that scans a plan, annotating each step within the scope of a norm with constraints stemming from that norm.

Each plan step is checked against the predicates specified in the specific norms (Ω^S), taking into account the role the agent adopts. If a step is within the scope of a norm, then the mechanism gradually assembles the constraints of the norms Γ_i, and annotates the plan step with them. If the norm is an obligation, the constraints are added as they appear in the norm, instantiated (or *customised*) to the substitutions σ, σ'. If the norm is a prohibition, the constraints are then *negated*; formally, $neg((\gamma_1, \ldots, \gamma_n)) = (neg(\gamma_1), \ldots, neg(\gamma_n))$, and each constraint can be negated as $neg(\tau > \tau') = (\tau \leq \tau')$, $neg(\tau < \tau') = (\tau \geq \tau')$, $neg(\tau \geq \tau') = (\tau < \tau')$, and so on. If the step is not in the scope of any norm, no constraints are added.

Once plan steps have been annotated, it is possible for an agent to check before executing each step if its execution violates a norm. However, it is inefficient to adopt a plan and execute it partially before discovering that the plan was not, in fact, desirable from the perspective of norm compliance. Fortunately, since the specific values of the variables within a plan are bound when a plan is instantiated, it is possible to determine at plan instantiation if any normative restriction applied to individual plan steps would be violated if the plan is adopted. In order to do this, we must make all annotations available for checking when the plan is instantiated so, at the end of each iteration over the steps of a plan, we collect the

annotations into a global plan annotation Γ', which is later used when *selecting* norm compliant plans.

EXAMPLE. The plan annotation mechanism, when applied to the plan introduced above, and using the specific norms shown previously, yields the following annotated plan:

$$\left\langle \begin{bmatrix} +level(X, medium), (high\_risk(X)), \\ isolate(X) \circ \top, \\ evacuate(X, Y) \circ \{Y \neq 3, Y \neq 6\}, \\ reroute(X, Z) \circ \{3 \leq Z \leq 5\} \\ \{Y \neq 3, Y \neq 6, 3 \leq Z \leq 5\} \end{bmatrix}, \right\rangle$$

We notice on the $evacuate$ step of the plan, the negated constraints of the specific norms arising from norm 1, shown with the substitutions applied. We also notice the $reroute$ step annotated with the constraints of the specific version of the obligation (also with the substitutions applied, and the mathematical expressions of the constraints simplified to improve visualisation). The annotated plan factors in the constraints of the active norms, making ν-BDI agents norm-aware.

D. Selection of Norm Annotated Plans

We now describe the process of selecting plans that comply with the constraints imposed by currently active norms. As we have seen, plans are annotated with constraints on the values that action variables may have when a plan is instantiated; in order to filter out non-compliant plan instances we simply verify their satisfiability with variable bindings for candidate plan instances. For example, if there is a plan containing an action $move(X, Y)$, and norm $\mathsf{O}_{A:R} move(X, Y) \circ \{X \leq 10 \wedge Y \leq 5\}$ is active, then instances of the plan with X bound to values greater than 10 should not be adopted.

In practice, violating situations are identified if the plan constraints become unsatisfiable after substitutions stemming from the plan instantiation are applied to the plan's annotations. In our example, if X is bound to 11, the annotation becomes $\{11 \leq 10 \wedge Y \leq 5\}$, which is not satisfiable. Now, in order to select compliant plans, the plan annotations created earlier, customised (via substitution σ) to the instantiation of the plan caused by event e, need to be satisfiable.

V. CONCLUSIONS AND RELATED WORK

In this paper we have described a new norm representation formalism, using constraints as means to *precisely* specify the target of normative stipulations. These constraints are used to determine specific plan instantiations that comply with active norms, thus narrowing the acceptable domains for operation. Based on this, we have described mechanisms that enable these plan instantiations to restrict behaviour in support of compliance, avoiding violating plans. Importantly, our work enables selective and incremental norm violation in a *controlled* manner in cases where goal achievement would not otherwise be possible, or where norms are deliberately ignored.

We have implemented these mechanisms within ν-BDI, extending a traditional BDI interpreter, such as dMARS [4]. However, our mechanisms are sufficiently generic to enable inclusion in *any* BDI interpreter and sufficiently detailed that implementation is straightforward. In addressing normative reasoning to this level of analysis, we have tackled various technical challenges posed by norm processing, such as the detection of activation and expiration conditions and the management of the norm life cycle between these two conditions, through the management of abstract and specific norms. Finally, we have shown the applicability of the mechanisms developed in an emergency evacuation scenario. In future work, we intend to refine the evacuation scenario as a testbed for our interpreter, and handle norm deadlines as well as normative conflicts.

Acknowledgement: This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] K. R. Apt. *From Logic Programming to Prolog*. Prentice-Hall, 1997.
[2] Michael E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.
[3] H. Bürckert. A resolution principle for constrained logics. *Artificial Intelligence*, (66):235–271, 1994.
[4] Mark d'Inverno, Michael Luck, Michael Georgeff, David Kinny, and Michael Wooldridge. The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System. *Autonomous Agents and Multi-Agent Systems*, 9(1–2):5–53, 2004.
[5] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.
[6] A. García-Camino, J.-A. Rodríguez-Aguilar, C. Sierra, and W. W. Vasconcelos. Constraint Rule-Based Programming of Norms for Electronic Institutions. *Autonomous Agents and Multiagent Systems*, 18(1):186–217, 2009.
[7] Peter Gärdenfors, editor. *Belief Revision*. Cambridge University Press, 2003.
[8] Davide Grossi, Huib Aldewereld, and Frank Dignum. Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *COIN II*, volume 4386 of *LNCS*, pages 101–114. Springer, 2007.
[9] François Félix Ingrand, Raja Chatila, Rachid Alami, and Frédéric Robert. PRS: A high level supervision and control language for autonomous mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 43–49, 1996.
[10] Joxan Jaffar, Michael J. Maher, Kim Marriott, and Peter J. Stuckey. The Semantics of Constraint Logic Programs. *J. Logic Programming*, 37(1-3):1–46, 1998.
[11] Andrew J. I. Jones and Marek Sergot. *On the characterisation of law and computer systems: the normative systems perspective*, pages 275–307. Wiley, 1993.
[12] Martin J. Kollingbaum and Tim J. Norman. Norm adoption in the NoA agent architecture. In *Proc. 2nd Int. Joint Conf. on Autonomous Agents & Multi-Agent Systems*. ACM, 2003.
[13] Alessio Lomuscio and Marek Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.
[14] Fabiola Lopez y Lopez, Michael Luck, and Mark d'Inverno. A normative framework for agent-based systems. In *Proc. 1st Int. Symp. on Normative Multi-Agent Systems*, 2005.
[15] Felipe Meneguzzi and Michael Luck. Norm-based behaviour modification in BDI agents. In *Proc. 8th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 177–184, 2009.
[16] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.
[17] Wamberto W. Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 9(2):124–152, 2009.
[18] G. H. von Wright. *An Essay in Deontic Logic and the General Theory of Action*. North-Holland, 1968.