Programming Norm-Aware Agents

Natasha Alechina
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
nza@cs.nott.ac.uk

Mehdi Dastani
Department of Information and Computing Sciences
Universiteit Utrecht
3584CH Utrecht, The Netherlands
M.M.Dastani@uu.nl

Brian Logan
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
bsl@cs.nott.ac.uk

ABSTRACT

Normative organisations provide a means to coordinate the activities of individual agents in multiagent settings. The coordination is realized at run time by creating obligations and prohibitions (norms) for individual agents. If an agent cannot meet an obligation or violates a prohibition, the organisation imposes a sanction on the agent. In this paper, we consider *norm-aware* agents that deliberate on their goals, norms and sanctions before deciding which plan to select and execute. A norm-aware agent is able to violate norms (accepting the resulting sanctions) if it is in the agent's overall interests to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable. Programming norm-aware agents in conventional BDI-based agent programming languages is difficult, as they lack support for deliberating about goals, norms, sanctions and deadlines. We present the norm-aware agent programming language N-2APL. N-2APL is based on 2APL and provides support for beliefs, goals, plans, norms, sanctions and deadlines. We give the syntax and semantics of N-2APL, and show that N-2APL agents are rational in the sense of committing to a set of plans that will achieve the agent's most important goals and obligations by their deadlines while respecting its most important prohibitions.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Languages, Theory

Keywords

Agent programming languages, Normative systems

1. INTRODUCTION

Normative organisations, e.g., [5], provide a means to coordinate the activities of individual agents in a multiagent system. In a normative organisation, coordination is realised at run time by creating obligations and prohibitions (norms) for individual agents. An obligation requires an agent to bring about a particular state of the environment by a specified deadline, while a prohibition requires the agent to avoid bringing about a particular state before a deadline. If an agent cannot meet an obligation or violates a prohibition, the organisation imposes a sanction on the agent.

In general, norms imposed on an agent by a normative organisation may conflict with the agent's existing goals. In such a situation, a rational agent must choose between its existing goals and the norms imposed by the organisation. We say an agent is *norm-aware* if it can deliberate on its goals, norms and sanctions before deciding which plan to select and execute. A norm-aware agent is able to violate norms (accepting the resulting sanctions) if it is in the agent's overall interests to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable. As an example, consider an agent that has agreed to review papers for a conference. The normative system in this case is the conference organisation, and the sanction for not discharging reviewing obligations by the review deadline may be reputational damage (e.g., being put on a blacklist). Let us further assume that the reputational damage for being late with reviews for an important conference such as AAMAS is greater than that incurred for being late with reviews for informal workshops. A norm-aware rational agent may still consider defaulting on its obligations to review for AAMAS if it acquires a more important goal with a tighter deadline, such as attending to some family emergency. Note that while we assume the severity of sanctions (and hence the priority or importance of obligations and goals) can be compared, their values are not necessarily commensurable in the sense that we cannot say whether being late with AAMAS reviews incurs the same sanction as being late with reviews for, e.g., two informal workshops.

There has recently been considerable work on programming frameworks for developing normative organisations [5, 14, 9]. Such frameworks are often designed to inter-operate with existing BDI-based agent programming languages, e.g., [2, 4]. However, programming norm-aware agents in conventional BDI-based agent programming languages remains difficult, as such languages typically lack support for deliberating about goals, norms, sanctions and deadlines.

In this paper we present a BDI-based agent programming language N-2APL for norm-aware agents. N-2APL extends 2APL [4] with support for normative concepts including obligations, prohibitions, sanctions, deadlines and durations. We give the syntax and operational semantics of N-2APL and explain how it supports norm-aware deliberation. We show that agents programmed in N-2APL are *norm-aware rational*, and that key assumptions underlying the design of N-2APL are necessary in the sense that if they are relaxed, a N-2APL agent is either no longer norm-aware rational, or the agent's deliberation about goals, norms and sanctions is intractable.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. NORMATIVE SYSTEMS

We conceive a multiagent system as consisting of a set of agents interacting with each other and with a shared environment. In normative multiagent systems, the interaction between agents and the environment is governed by a normative exogenous organisation, which is specified by a set of conditional norms with their associated deadlines and sanctions. Individual agents decide which action to perform in the environment after which the state of the environment changes. Subsequently, the organisation evaluates the environmental changes with respect to the conditional norms to: determine any obligations to be fulfilled or prohibitions that should not be violated by the agents (termed detached obligations and prohibitions); determine any violations based on the deadlines of the detached obligations and prohibitions; and impose any corresponding sanctions. The role of the exogenous organisation is thus to continuously 1) monitor the behaviour of agents, 2) evaluate the effects they bring about in the environment, 3) determine norms that should be respected, 4) check if any norm is violated, and 5) take necessary measures (i.e., imposing sanctions) when norms are violated. This continuous process is often implemented by a so-called control cycle [5].

A conditional obligation is expressed as a tuple

$$\langle c, O(\iota, o), d, s\rangle$$

with the intuitive reading "if condition c holds in the current state of the environment then there is an obligation for agent ι to establish an environment state satisfying o before deadline d, otherwise agent ι will be sanctioned by updating the environment with s". In the conference reviewing example, a possible norm is to return reviews for a paper assigned to a reviewer. An instance of such a conditional norm could be represented as:

$$\langle collect\ \&\ p_{id}Ar_i, O(r_i, p_{id}Rev), notify, BLr_i\rangle$$

which indicates that when a conference is in the review collection phase ($collect$) and paper p_{id} is assigned to reviewer r_i ($p_{id}Ar_i$), then the reviewer r_i is obliged to return the review of paper p_{id} ($p_{id}Rev$) before the notification phase ($notify$) starts. Violating this norm results in the reviewer being put on a blacklist (BLr_i), damaging its reputation.

A conditional prohibition is expressed as a tuple

$$\langle c, F(\iota, p), d, s\rangle$$

with the intuitive reading "if condition c holds in the current state of the environment, then it is forbidden for agent ι to establish an environment state satisfying p before deadline d, otherwise sanction s will be imposed." Unlike obligations, where a sanction is incurred once if the obligation is not discharged by the deadline, in the case of prohibitions, the agent incurs a sanction each time the prohibition is violated. For example, if it prohibited to submit a paper longer than 8 pages, the agent will incur a sanction (e.g., the paper being rejected without review) each time it submits a paper longer than 8 pages. In what follows, we consider the simpler case of prohibitions without deadlines (i.e., prohibitions with an indefinite deadline or with $\perp$ as deadline).

As our focus is on an agent's decision problem when operating in a normative multiagent system, we ignore the working of the normative exogenous organisation, and simply assume an additional step in the control cycle through which the organisation broadcasts detached obligations and prohibitions in the form of events (i.e., a set of normative events) to each agent to whom the norms are directed. A detached obligation event broadcast to agent ι has the form:

$$obligation(\iota, o, d, s)$$

with the intuitive reading "agent ι is obliged to establish an environment state satisfying o before deadline d, otherwise it will be sanctioned by updating the environment with s". For example, when a paper is assigned to a reviewer during the collection phase, the organisation generates and sends the following (singleton) set of detached obligations to the corresponding reviewer r_i:

$$\{obligation(r_i, p_{id}Rev, notify, BLr_i)\}$$

We assume that deadlines associated with detached obligations can be mapped to real time values expressed in some appropriate units that specify the time by which the obligation should be discharged. For example, *notify* might map to the real time value "5pm on Friday" which specifies the time by which the review should be returned. Similarly, a detached prohibition event broadcast to agent ι has the form:

$$prohibition(\iota, p, s)$$

with the intuitive reading "the agent ι is prohibited from establishing an environment state satisfying p, otherwise it will be sanctioned by updating the environment with s"

Of course, obligations or prohibitions from a normative organisation may conflict with an agent's goals or with other obligations or prohibitions it has already received (possibly from another normative organisation). For example, if paper reviewing and family emergencies cannot be attended to concurrently, the agent needs to schedule its intentions in some order, such as: first deal with the emergency, then review paper 1, then review paper 2, etc. Some schedules will be better than others. For example, if paper 1 has an earlier review deadline than paper 2, and the agent still has sufficient time after dealing with the emergency to review paper 1, and (after reviewing paper 1) it still has time to review paper 2, the schedule above may be optimal. On the other hand, if the agent does not have enough time left to review both papers by their deadlines and reviewing paper 2 is a more important obligation (incurs a greater sanction), then rationally it should review paper 2 rather than paper 1 next.

We assume that an agent has a preference or priority ordering over goals and sanctions that determines whether it is more important (from the point of view of the agent) to, e.g., achieve a goal g or to avoid the sanction s associated with an obligation or prohibition. In the case of goals, the priority indicates the importance of *achieving* the goal state, while in the case of sanctions, the priority indicates the importance of *avoiding* the sanction state, i.e., sanctions that entail a smaller penalty for the agent will have lower priority. Sanctions thus induce an order on obligations and prohibitions: the priority of an obligation or prohibition is determined by the priority of the sanction that would be incurred if the obligation is not discharged by its deadline or the prohibition violated.[1] We assume that an agent's preferences over goals and sanctions are ordered on an ordinal scale, i.e., that it is always preferable to achieve a higher priority goal (or avoid a high priority sanction) than to achieve any number of lower priority goals (or avoid any number of lower priority sanctions). (As we show in section 5, if this is not the case, the agent's deliberation about norms is intractable.)

If the agent's goals and obligations are not jointly achievable or its goals cannot be achieved without violating one or more prohibitions, the agent uses the relative priority of goals and sanctions to determine which goal(s) to drop or which sanction(s) to incur. Following BOID [3], we will consider special cases of the priority ordering and define them as specific agent types. For example, a

[1]Note that we do not assume that obligations are preferred or desirable states for agent; rather the agent is motivated by the avoidance of sanctions rather than the achievement of obligations.

social agent can be characterized by a priority ordering that prefers all obligations to its goals and a selfish agent can be characterized by a priority ordering that prefers its goals over its obligations.

If the agent's goals and obligations are not jointly achievable, a norm-aware rational agent should schedule its intentions so as to achieve goal and obligation states with highest priority. Programming norm-aware rational agents is non-trivial, as it depends on the deadlines of the goals and obligations, the plans the agent has to achieve its goals and obligations, whether its plans violate any prohibitions, the expected execution time of the agent's plans and the extent to which the plans can be executed in parallel.

3. THE LANGUAGE N-2APL

In this section we present N-2APL, an agent programming language for norm-aware rational agents. N-2APL is a modification of the agent programming language 2APL [4] which supports normative concepts including obligations, prohibitions, sanctions, deadlines and durations. We first briefly present 2APL, focusing on those elements modified in N-2APL, before describing the extended programming constructs of N-2APL and how it supports norm-aware deliberation.

3.1 2APL: a Brief Summary

2APL is a BDI-based agent programming language that allows the implementation of agents in terms of cognitive concepts such as beliefs, goals and plans. A 2APL agent program specifies an agent's initial beliefs, goals, plans, and the reasoning rules it uses to select plans (`PG-rules`), to respond to messages and events (`PC-rules`), and to repair plans whose executions have failed (`PR-rules`). The initial beliefs of an agent includes the agent's information about itself and its surrounding environment. The initial goals of an agent consists of formulas each of which denotes a situation the agent wants to realize (not necessarily all at once). The initial plans of an agent consists of tasks that an agent should initially perform.

In order to achieve its goals, an 2APL agent adopts plans. A plan consists of basic actions composed by sequence, conditional choice, conditional iteration and non interleaving operators. The non interleaving operator, $[\pi]$ where π is a plan, indicates that π is an *atomic* plan, i.e., the execution of π should not be interleaved with the execution of any other plan. Basic actions include external actions (which change the state of the agent's environment); belief update and goal adopt actions (which change the agent's beliefs and goals), and abstract actions (which provide an abstraction mechanism similar to procedures in imperative programming).

Planning goal rules allow an agent to select an appropriate plan given its goals and beliefs. A planning goal rule $\langle pgrule \rangle$ consists of three parts: the head of the rule, the condition of the rule, and the body of the rule. The body of the rule is a plan that is generated when the head (a goal query) and the condition (a belief query) of the rule are entailed by the agent's goals and beliefs, respectively. Procedure call rules are used to select a plan for an abstract action and to handle external events. Plan repair rules are used to revise plans whose executions have failed.

3.2 N-2APL Extensions and Restrictions

To support norm-aware agents, N-2APL extends some key constructs of 2APL and restricts or changes the semantics of others. We briefly describe these changes below. The syntax of N-2APL is shown in Figure 1 in EBNF notation. Programming constructs in bold are exactly the same as in 2APL. Due to lack of space, we omit the detailed specification of these programming constructs, which can be found in [4].

Beliefs in N-2APL are exactly the same as in 2APL and consist of Horn clause expressions. Goals in 2APL may be conjunctions of positive literals. In N-2APL we restrict goals to single atoms and extend their syntax to include optional deadlines. A *deadline* is a real time value expressed in some appropriate units which specifies the time by which a goal should be achieved. We write a goal g with a deadline d as $g : d$. If no deadline is specified for a goal as part of the agent's program, we assume a deadline of infinity.

In N-2APL, non-atomic plans are the same as in 2APL and consist of basic actions composed by sequence, conditional choice, and conditional iteration operators. However in N-2APL we change the interpretation of the non interleaving operator: $[\pi]$ indicates that the execution of π should not be interleaved with the execution of other *atomic* plans (rather than not interleaved with the execution of *any* other plan as in 2APL). In N-2APL, atomic plans are assumed to contain basic actions that may interfere only with the basic actions in other atomic plans, e.g., belief update actions that update the same belief(s), or external actions that change the position of the agent etc. For example, for a particular agent, reviewing a paper may require the agent's undivided attention and cannot be executed in parallel with reviewing another paper. However other plans, such as having lunch and taking a train can be executed in parallel (the agent can have a sandwich on the train while reviewing the paper). As illustrated by the example, in N-2APL, we also allow external actions in different non-atomic plans to be executed in parallel, rather than interleaved as in 2APL (see section 4 for details). Lastly, we restrict the scope of the non interleaving operator such that non-atomic plans cannot contain atomic sub-plans, either directly or through the expansion of an abstract action, i.e., plans to achieve top-level goals are either wholly atomic or non-atomic. As we show in section 5, these changes are necessary for the agent's deliberation about norms to be tractable.

We extend the syntax of plans in the body of a PG rule to include an optional field specifying the time required to execute the plan proposed by the PG rule. In N-2APL, a PG rule has the form:

$$\kappa \leftarrow \beta \mid \pi : t$$

where κ is a goal query, β is a belief query, π is a plan and t is the time required to execute π. We assume that the time required to execute a plan is primarily determined by the time required to execute the external actions it contains, i.e., that the amount of time required to execute internal actions (belief update, goal adopt, abstract actions etc.) is small compared to the time required to execute external actions. The problem of determining t for a plan π therefore reduces to the problem of determining the sequence of external actions that will be executed, and estimating the time required to execute each of these external actions. For simplicity, we assume that the time required to execute each plan π is fixed and known in advance.

As explained in section 2, the creation of an obligation or prohibition by an external organisation causes an event to be sent to the agent. An obligation event, represented as $obligation(\iota, o, d, s)$, specifies the time d by which the obligation o must be discharged, i.e., its deadline, and the sanction, s, that will be applied if the obligation is not discharged by the deadline. A prohibition event, represented as $prohibition(\iota, p, s)$, specifies a prohibition p that must not be violated and the sanction s that will be applied if execution of the agent's plans violates the prohibition. Obligations and prohibitions are added to the agent's goal and event bases, respectively. In particular, an obligation is adopted as a goal $o : d$ with priority corresponding to (the importance of avoiding) s, and a prohibition is represented by a prohibition event $prohibition(p, s)$ where the priority of p corresponds to (the importance of avoiding)

⟨*Agent_Prog*⟩ = ["Beliefs:" { ⟨**belief**⟩ }] ,
["Goals:" ⟨*goals*⟩] ,
["Plans:" ⟨*plans*⟩] ,
["PG-rules:" { ⟨*pgrule*⟩ }] ,
["PC-rules:" { ⟨**pcrule**⟩ }]
["PR-rules:" { ⟨**prrule**⟩ }]
⟨*goals*⟩ = ⟨*goal*⟩ { "," ⟨*goal*⟩ } ;
⟨*goal*⟩ = ⟨*atom*⟩ ":" ⟨*deadline*⟩;
⟨*pgrule*⟩ = ⟨*goalquery*⟩ "<-" ⟨*belquery*⟩ "|" ⟨*plan*⟩ ":" ⟨*duration*⟩;
⟨*goalquery*⟩ = ⟨*goalquery*⟩ "and" ⟨*goalquery*⟩ | ⟨*goalquery*⟩ "or" ⟨*goalquery*⟩ | "(" ⟨*goalquery*⟩ ")" | ⟨*atom*⟩ ;
⟨*belquery*⟩ = ⟨*belquery*⟩ "and" ⟨*belquery*⟩ | ⟨*belquery*⟩ "or" ⟨*belquery*⟩ | "(" ⟨*belquery*⟩ ")" | ⟨*literal*⟩ ;
⟨*plan*⟩ = ⟨*atomic-plan*⟩ | ⟨**non-atomic-plan**⟩;
⟨*atomic-plan*⟩ = "[" ⟨**non-atomic-plan**⟩ "]";
⟨*sanction*⟩ = ⟨*atom*⟩;
⟨*deadline*⟩ = ⟨*time*⟩;
⟨*duration*⟩ = ⟨*int*⟩;

Figure 1: EBNF syntax of N-2APL.

the sanction s (see section 4). We assume the programmer provides a binary relation $pref(x, y)$ where x, y may be goals, obligations or prohibitions that returns true if the goal (sanction) x has higher priority than the goal (sanction) y, and that the order induced by $pref$ is stable. Finally, we assume that it is possible to define a function $effects(\pi)$ which returns the set of literals appearing in the postconditions of all external actions in π. A plan π violates a prohibition p iff $p \in effects(\pi)$, i.e., if executing the plan would cause p to become true.

3.3 Norm-aware Deliberation

In determining which plan to adopt for a goal, a norm-aware agent must take into account (and possibly revise) plans to which it is currently committed. In addition it must decide when each plan to which it is committed should be executed, i.e., it must schedule the execution of its plans. Informally, a schedule is an assignment of a start or next execution time to a set of plans which ensures that: all plans complete by their deadlines, at most one atomic plan executes at any given time, and where the goals achieved and the prohibitions avoided are of the highest priority.[2]

To define a schedule, we first define a set of feasible plans. A set of plans $\Pi = \{\pi_1, \ldots, \pi_n\}$ is *feasible* iff:

1. it is possible for each plan to complete execution before its deadline, that is, for each plan $\pi_i \in \Pi$
$$ne(\pi_i) + et(\pi_i) - ex(\pi_i) \leq dl(\pi_i)$$
where $ne(\pi_i)$ is the time at which π_i will next execute, $et(\pi_i)$ is the time required to execute π_i, $ex(\pi_i)$ is the time π_i has spent executing up to this point and $dl(\pi_i)$ is the deadline for π_i (we assume that all plans π_i complete by $et(\pi_i)$ and hence $et(\pi_i) - ex(\pi_i)$ is always non-negative);

2. if π_i is an atomic plan, then no other atomic plan is scheduled to execute concurrently with π_i, namely the set $\{\pi_j \in \Pi \setminus \{\pi_i\} \mid (ne(\pi_i) < ne(\pi_j) + et(\pi_j)) \wedge (ne(\pi_j) < ne(\pi_i) + et(\pi_i))\}$ contains no atomic plans; and

3. if π_i is a currently executing atomic plan whose deadline has not passed, for any plan $\pi_j \in \Pi \setminus \{\pi_i\}$, $ne(\pi_j) > now + et(\pi_i) - ex(\pi_i)$, i.e., an atomic plan cannot preempt a currently executing atomic plan.

Feasibility determines which sets of plans can be executed by their deadlines without violating atomicity constraints.

The agent commits to a feasible set of plans that is preference-maximal. If it is not possible to execute all plans by their deadlines it will drop plans that achieve goals of lower priority in preference to plans which achieve goals of higher priority. Similarly, it will drop plans that violate prohibitions of higher priority than the goal achieved by the plan.

To make this precise, we define a *preference-maximal* set of plans as follows. Consider a set of plans $\Pi = \{\pi_1, \ldots, \pi_n\}$ and prohibitions $\Delta = \{p_1, \ldots, p_m\}$. $\Gamma \subseteq \Pi$ is preference-maximal (with respect to Π and Δ) iff:

1. Γ is feasible;

2. $\forall \pi_i \in \Pi$ such that $\pi_i \notin \Gamma$, either

   - $\{\pi_i\}$ is infeasible, or
   - $\exists \Gamma' \subseteq \Gamma$: the minimal priority of a plan in Γ' is greater than or equal to the priority of π_i, and $\{\pi_i\} \cup \Gamma'$ is infeasible; or
   - $\exists p_j \in \Delta, p_j \in effects(\pi_i)$ and the priority of p_j is greater than or equal to the priority of π_i

Intuitively, this definition describes a subset of Π which is 'maximally feasible' (no more plans from Π can be added if the plans are to remain feasible) and moreover, plans in $\Pi \setminus \Gamma$ cannot be scheduled together with some subset of Γ that contains plans(s) of higher priority or they violate a prohibition in Δ of the same or higher priority.

A *schedule* is a preference-maximal set of plans together with their start (next execution) times.

The agent uses a *schedulability criterion* when deliberating about which plan to adopt for a goal, and which plans to drop. In order to decide if a PG-rule
$$\kappa \leftarrow \beta \mid \pi : t$$
is applicable, we check that: $\gamma \models_g \kappa$ (i.e., that $g\,\tau \models \kappa$ for some $g : d \in \gamma$ and substitution τ), $\sigma \models \beta$ and $\exists \Gamma \subseteq \Pi$ such that $\Gamma \cup \{\pi\}$ is preference-maximal relative to Π, Δ, where γ is the agent's goal base, σ is the agent's belief base, Π is the agent's plan base and Δ is the agent's prohibitions. The first two conditions are straightforward and simply check that the agent has a plan which will achieve

[2] In what follows, in the interests of brevity we shall often refer to the 'deadline' and 'priority' of a plan π rather than the deadline of the goal achieved by π or the priority of the goal achieved or the sanction avoided by executing π.

the goal and that plan is applicable in the current belief context. The third condition is more complex: it checks whether adopting the plan π is rational for the agent. Adopting π is rational if π together with some subset Γ of the agent's plan base is feasible, and any plans $\pi' \in \Pi \backslash \Gamma$ that must be dropped in order to schedule π by its deadline have strictly lower priority than the goal or obligation achieved by π.

A N-2APL agent thus adopts an open-minded commitment strategy — at any given point in its execution its plan base comprises a preference-maximal set of plans. Observe that if an agent is 'social' and orders its obligations before any of its own goals, then all obligations will be discharged by the agent provided it has feasible plans to achieve them. On the other hand, selfish agents which prioritize their own goals, may incur (in the worst case, when they do not attend to any of the obligations and violate all prohibitions) all the sanctions possible in the normative system.

3.4 Scheduling Algorithm

Algorithm 1 Scheduling Algorithm

```
1: function SCHEDULE(Π, Δ)
2:     Γ_s := ∅, Γ_p := ∅,
3:     for all π ∈ Π in descending order of priority do
4:         V := effects(π) ∩ Δ
5:         if ¬atomic(π) then
6:             if now + rt(π) ≤ dl(π) ∧
7:                pr(π) ≥ argmax pr(p), p ∈ V then
8:                 ne(π) := now
9:                 Γ_p := Γ_p ∪ {π}
10:            end if
11:        else
12:            if executing(π) then
13:                ne(π) := now
14:                Γ'_s := Γ_s
15:            else
16:                t := now
17:                Γ'_s := ∅
18:                for all π' ∈ Γ_s do
19:                    if dl(π') ≤ dl(π) then
20:                        Γ'_s := Γ'_s ∪ {π'}
21:                        t := max(ne(π') + rt(π'), t)
22:                    else
23:                        ne(π') := ne(π') + rt(π)
24:                        Γ'_s := Γ'_s ∪ {π'}
25:                    end if
26:                end for
27:                ne(π) := t
28:            end if
29:            if ∀π_i ∈ Γ'_s ∪ {π}: now + Σ_{ne(j)≤ne(i)} rt(π_j) ≤ dl(π_i) ∧
30:               pr(π) ≥ argmax pr(p), p ∈ V then
31:                Γ_s = Γ'_s ∪ {π}
32:            end if
33:        end if
34:    end for
35:    return Γ_p ∪ Γ_s
36: end function
```

Scheduling in N-2APL is pre-emptive in that the adoption of a new plan π may prevent previously scheduled plans with priority lower than π (including currently executing plans) being added to the new schedule. Plans that would exceed their deadline are dropped. In the case of obligations, a sanction will necessarily be incurred, so it is not rational for the agent to continue to attempt to discharge the obligation. In the case of goals, it is assumed that the deadline is hard, and there is no value in attempting to achieve the goal after the deadline.

The scheduling algorithm is shown in Algorithm 1. $ne(\pi)$ is the time at which π_i will next execute, $ex(\pi)$ is the time π_i has spent executing up to this point, $dl(\pi)$ is the deadline for π, and $rt(\pi) = et(\pi) - ex(\pi)$ is the remaining execution time of π.

Non-atomic and atomic plans are scheduled separately in Γ_p and Γ_s respectively. The set of candidate plans is processed in descending order of priority. For each plan π, if π is non-atomic an attempt is made to schedule it in parallel with other non-atomic plans in Γ_p (lines 6–10). To determine feasibility for non-atomic plans it is sufficient to check that the plan can be executed by its deadline. If the plan is atomic, it is added to the schedule Γ_s if it can be inserted into the schedule in deadline order while meeting its own and all currently scheduled deadlines (lines 12–32). A set of atomic plans is feasible iff they can be scheduled earliest deadline first [10]. If a non-atomic or atomic plan violates a prohibition of higher priority than the plan, the plan is dropped. The resulting schedule can be computed in polynomial time (in fact, quadratic time) in the size of Π, and (as we show in section 5) is preference-maximal.

4. OPERATIONAL SEMANTICS

In this section, we sketch the operational semantics of N-2APL in terms of a transition system. Each transition transforms one configuration into another and corresponds to a single computation/execution step. In the following subsections, we first present the configuration of individual N-2APL agent programs (henceforth agent configuration) then the configuration of multiagent system programs (henceforth multiagent system configuration), before finally presenting transition rules from which possible execution steps (i.e., transitions) for both individual agents as well as multiagent systems can be derived.

4.1 N-2APL Configuration

The configuration of an individual agent consists of its identifier, beliefs, goals, prohibitions, planning goal rules, procedure call rules, plans, events, and a preference ordering on goals and sanctions. Each plan is associated with the goal and practical reasoning rule that gave rise to the plan (in order to avoid redundant applications of practical reasoning rules, e.g., to avoid generating multiple plans for one and the same goal). It should be noted that the belief base and each goal in the goal base are consistent as only positive atoms are used to represent them.

DEFINITION 1 (AGENT CONFIGURATION). *The configuration of a N-2APL agent is defined as $A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi, \succ, PG, PC, PR \rangle$ where ι is the agent's identifier, σ is a set of belief expressions $\langle belief \rangle$ representing the agent's belief base, γ is a list of goal expressions $\langle goal \rangle$representing the agent's goal base, Π is the agent's plan base consisting of a set of plan entries $(\langle plan \rangle, \langle goal \rangle, \langle pgrule \rangle)$ representing the agent's plans together with their next execution times, ξ is the agent's event base containing also elements of the form $prohibition(p, s)$, $\succ$ is the preference ordering, PG is the set of planning goal rules, PC is a set procedure call rules, and PR is a set of plan repair rules.*

Since the agent's practical reasoning rules and preference ordering do not change during an agent's execution, we do not include them in the agent's configuration and use $A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \rangle$ to denote an agent's configuration.

The configuration of a multiagent system is defined in terms of the configuration of individual agents and the state of their organisation. The state of the agents' organisation is a set of facts that hold in that organisation.

DEFINITION 2 (MULTIAGENT SYSTEM CONFIGURATION). *Let A_ι be the configuration of agent ι and let χ be the state of the agents' organisation. The configuration of a N-2APL multiagent system is defined as $\langle A_1, \ldots, A_n, \chi\rangle$.*

The initial configuration of a multiagent system is determined by its corresponding multiagent system program and consists of the initial configuration of its individual agents (determined by their corresponding N-2APL programs) and the initial state of their organisation.

DEFINITION 3 (INITIAL CONFIGURATION). *Let ι be the identifier of an agent that is implemented by a N-2APL program. Let σ be the set of $\langle belief\rangle$-expressions specified in the N-2APL program and γ be the list of $\langle goal\rangle$-expressions from the same program. Then, the initial configuration of agent ι is defined as tuple $\langle \iota, \sigma, \gamma, \emptyset, \emptyset\rangle$. Let also χ be a set of facts and $A_1, \ldots, A_n$ be the initial configurations of agents $1, \ldots, n$ that are specified in the multiagent system program. The initial configuration of the multiagent systems is defined as tuple $\langle A_1, \ldots, A_n, \chi\rangle$.*

4.2 Transition Rules

The execution of a N-2APL multiagent program modifies its initial configuration by means of transitions that are derivable from the transition rules given below. Due to lack of space, we do not provide transition rules for the execution of plans, see e.g., [4].

4.2.1 Receiving Detached Norms

As explained in section 2, a normative organisation can broadcast an obligation or a prohibition event to a specific agent. We assume transition rules from which a transition of a normative organisation is derivable, i.e., we assume transition rules that can be used to derive transitions $\chi \xrightarrow{n\text{-}event} \chi'$, where *n-event* is an event such as $obligation(\iota, o, d, s)$ or $prohibition(\iota, p, s)$. Such transition rules specify under which conditions an obligation or a prohibition should be issued for a specific agent. The study of such conditions is out of the scope of this paper and can be found in, e.g., [5]. The following transition rule allows a normative organisation to broadcast normative events (e.g, $obligation(\iota, o, d, s)$ or $prohibition(\iota, p, s)$) and ensures that the events are delivered to the appropriate agent. An obligation event is added to the agent's goal base and a prohibition event is added to its event base.

$$\frac{\chi \xrightarrow{n\text{-}event} \chi'}{\langle A_0, \ldots, A_\iota, \ldots, A_n\rangle \longrightarrow \langle A_0, \ldots, A'_\iota, \ldots, A_n\rangle} \quad (1)$$

where
$A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi\rangle$,
$A'_\iota = \langle \iota, \sigma, \gamma \cup \{o{:}d\}, \Pi, \xi, \succ'\rangle$ if $n\text{-}event = obligation(\iota, o, d, s)$,
$A'_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \cup \{prohibition(p, s)\}, \succ'\rangle$ if $n\text{-}event = prohibition(\iota, p, s)$.

4.2.2 Planning Goal Rules

A N-2APL agent generates plans by applying PG-rules of the form $\kappa \leftarrow \beta \mid \pi : t$. An agent can apply one of its PG-rules $\kappa \leftarrow \beta | \pi : t$, if κ is entailed (with some substitution τ_1) by one of the agent's goals, namely by some g such that $g : d \in \gamma$, β is entailed (with substitution $\tau_1\tau_2$) by the agent's belief base, and there is no plan in the plan base that has been generated (and perhaps partially executed) by applying the same PG-rule to achieve the same goal. Applying the PG-rule $\kappa \leftarrow \beta \mid \pi : t$ attempts to add $\pi\,\tau_1\tau_2$ with deadline d and execution time t to the agent's plan base.

$$\frac{\begin{array}{c}\exists\,(g{:}d) \in \gamma : g \models_g \kappa\,\tau_1 \;\&\; \sigma \models \beta\,\tau_1\tau_2 \\ \;\&\; \neg\exists \pi' \in P : (\pi', g{:}d, (\kappa\,\tau_1 \leftarrow \beta \mid \pi{:}t)) \in \Pi\end{array}}{\langle \iota, \sigma, \gamma, \Pi, \xi\rangle \longrightarrow \langle \iota, \sigma, \gamma, \Pi', \xi\rangle} \quad (2)$$

where τ_1, τ_2 are substitutions, P is the set of all possible plans and $\Pi' = \text{SCHEDULE}(\Pi \cup \{(\pi\,\tau_1\tau_2,\ g : d,\ (\kappa\,\tau_1 \leftarrow \beta \mid \pi : t))\}, prohibitions(\xi))$. Here, SCHEDULE is defined as in Algorithm 1 and *prohibitions* is a function that takes a set of events and returns all prohibition events in that set.

4.2.3 Concurrent Plans

The applications of planning goal rules may generate both atomic and non atomic plans. External actions in different non-atomic plans can be executed in parallel. Atomic plans are executed in sequence, rather than in parallel. Actions within each plan are executed in strict sequence, i.e., the next action in the plan is not executed until the previous action has completed execution or failed. We say a plan is *executable* if its next execution time is *now* and the previous action in the plan has successfully completed execution, otherwise it is not executable (i.e., an action initiated at a previous deliberation cycle is still executing, or has failed, or the plan is atomic and scheduled for execution at a later time). Execution of internal actions is assumed to occur in the main interpreter thread, whereas execution of external actions is assumed to occur in separate threads (otherwise the actual execution time of a plan π would bear little relation to its expected execution time, $et(\pi)$) .

An agent executes its plans concurrently by interleaving the execution of the next action (or the initiation of the execution of the action in a separate thread in the case of external actions) of all executable plans whose next execution time is *now*

$$\frac{\langle \iota, \sigma, \gamma, \rho, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \rho', \xi'\rangle}{\langle \iota, \sigma, \gamma, \Pi, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \Pi', \xi'\rangle} \quad (3)$$

where ρ is executable and $\Pi' = (\Pi \setminus \rho) \cup \rho'$.

5. NORM-AWARE RATIONALITY

In this section, we justify the adoption of a preference-maximal set of plans as an appropriate standard of rationality for a norm-aware agent. We show that key assumptions underlying the design of N-2APL are necessary in the sense that if they are relaxed, a N-2APL agent is either no longer a norm-aware rational agent, or its deliberation about goals, norms and sanctions is intractable.

In what follows, we assume that the agent's plans have a positive expected execution time and a deadline, and that atomic plans are scheduled on a single processor. We also assume that plans and prohibitions have priorities, which are either ordered by a total preference pre-order, or have numerical values. The definition of a feasible set of plans is given in section 3.3. Note that in order to be able to establish whether a set of plans is feasible, the agent needs to know expected execution times of the plans.

We call a set of plans *optimal* if it is feasible and has maximal utility; namely, if the preferences over goals and sanctions are ordered by a preference pre-order, then it contains the highest number of the high priority plans; if priorities are numerical, then their sum is maximal.

DEFINITION 4. *An agent is* perfectly norm-aware rational *if it commits to an optimal set of plans.*

The problem of finding an optimal schedule is NP-complete (e.g., the special case in which all tasks have the same deadline can be reformulated as a 0-1 knapsack problem [7]).

Since the problem of scheduling an optimal set of plans is not tractable, we have the following theorem:

THEOREM 1. *A perfectly norm-aware rational agent cannot have a tractable deliberation procedure.*

In this paper, we assume a weaker definition of norm aware rationality. This definition relies on the notion of a preference-maximal set of plans given in Section 3.3 and requires that preferences are ordered on an ordinal scale. Note that in this case every optimal set of plans is preference-maximal, but not vice versa. For example, for three atomic plans with the same priority, π_1, π_2 and π_3, such that π_1 is not feasible with either of π_2 or π_3 and $\{\pi_2, \pi_3\}$ is feasible, the only optimal set is $\{\pi_2, \pi_3\}$. However, $\{\pi_1\}$ is a preference-maximal set of plans.

DEFINITION 5. *An agent is* norm-aware rational *if it commits to a preference-maximal set of plans.*

The set of goals achieved by a *norm-aware rational* agent is determined by its program. If the program is such that the belief contexts of PG rules are disjoint, then the set of goals achieved by the successful execution of a preference-maximal set of plans is also 'preference-maximal', in the sense that the agent will only fail to achieve a goal if it has no applicable plan for a goal, or the plan to achieve the goal is not feasible together with the plans to achieve goals of the same or higher priority.

THEOREM 2. *A N-2APL agent is norm-aware rational and its deliberation procedure is tractable.*

PROOF. We show that the N-2APL scheduling algorithm returns a preference-maximal schedule; tractability is obvious. The N-2APL scheduling algorithm builds two separate schedules, a parallel and a sequential one. The set of plans Γ_p in the parallel schedule contains all non-atomic plans which are individually feasible (the time remaining to their deadline is greater than their expected execution time) and do not violate prohibitions of higher priority. This set of plans is clearly maximal given the prohibitions. Note that the feasibility of Γ_p is not affected by the membership of the sequential schedule Γ_s and vice versa.

For the sequential schedule, the algorithm constructs a sequence of sets starting with $\Gamma_0 = \emptyset$, and sets Γ_i to be $\Gamma_{i-1} \cup \{\pi_i\}$, if $\Gamma_{i-1} \cup \{\pi_i\}$ is feasible in deadline order, or Γ_{i-1} otherwise. The last set Γ_n is Γ_s. By construction, Γ_s is a feasible set of plans. Γ is also clearly a maximally feasible subset of Π: there is no atomic $\pi \in \Pi$ such that $\pi \notin \Gamma_s$ and $\Gamma_s \cup \{\pi\}$ is feasible. To prove that it is preference-maximal, let $\pi_i \in \Pi$, $\{\pi_i\}$ feasible, and $\pi_i \notin \Gamma_s$. We need to show that π_i is incompatible with some subset of Γ_s which contains only plans of the same or higher priority, or is incompatible with a higher priority prohibition. Since the plans are added to Γ_s in descending order of priority, when π_i is considered and found incompatible with Γ_{i-1}, the priority of π_i is at most the lowest priority in Γ_{i-1}. □

In the rest of this section, we show that the assumptions we made concerning the normative system and the agent programming language semantics are essential to guarantee tractability of a norm aware rational agent's deliberation.

THEOREM 3. *If a normative system has prohibitions with real-time deadlines, then a norm-aware rational agent cannot have a tractable deliberation procedure.*

PROOF. If prohibitions have real-time deadlines, the scheduling problem is equivalent to the 'sequencing with release times and deadlines' problem (SRTD), which is known to be strongly NP-complete [7]. SRTD is intractable in the sense that it admits no bounded approximation computable in time polynomial in the problem size and the bound (unless $P = NP$). □

N-2APL also places certain restrictions on the syntax of 2APL programs, and changes the semantics of key constructs such as the non interleaving (atomic) operator. If these assumptions are relaxed, the agent is no longer norm-aware rational or deliberation about norms is intractable. We now make these assumptions precise.

THEOREM 4. *If an agent's plans may have atomic sub-plans the agent is not norm-aware rational, or its deliberation procedure is intractable.*

PROOF. Note that if an agent's plans may contain atomic sub-plans $\pi = \pi_1; [\pi_2]; \pi_3; [\pi_4]$, the agent must schedule the atomic sub-plans $[\pi_2]$ and $[\pi_4]$ together with its other atomic plans, but subject to the constraint that the preceding non-atomic sub-plans π_1 (in the case of $[\pi_2]$) and π_2 (in the case of $[\pi_4]$) of π have finished executing. If information about the execution times of each atomic and non-atomic sub-plan is not available, the agent is not norm-aware in the sense that it may commit to plans that it subsequently has to abandon, even if its goals and norms do not change. In effect, the agent is unable to tell when the execution of an atomic sub-plan may have to be scheduled, and so may adopt a non preference-maximal schedule. If, on the other hand, we assume that information about the execution times of each plan segment is available (and is used in scheduling), the scheduling problem is again equivalent to SRTD and is intractable. □

THEOREM 5. *If atomic plans cannot be executed in parallel with non-atomic plans, an agent is not norm-aware rational or its deliberation procedure is intractable.*

PROOF. If we adopt the 2APL semantics for the non interleaving operator,[3] i.e., the execution of an atomic plan should not be interleaved with the execution of any other plan, we again get a combinatorial scheduling problem because non-atomic plans can be split in various groups depending on their end times. This problem reduces to the batch scheduling problem which is NP-hard [12]. □

THEOREM 6. *If external actions cannot be executed in parallel, an agent is not norm-aware rational or its deliberation procedure is intractable.*

PROOF. If external actions are interleaved, the expected execution time of a plan is dependent on the other plans in the agent's schedule. If the execution times of each external action in a plan are not known, the resulting schedule is not guaranteed to be preference-maximal. If the execution time of each external action are known (and is used in scheduling), the scheduling problem is again reducible to SRTD, for both atomic and non-atomic schedules. □

6. RELATED WORK

Our notion of norm-awareness is related to, e.g., [14], where it is argued that the ability of agents to reason about the norms of an organisation in which they operate is crucial for their decisions to enter and leave organisations or to respect/violate norms.

There has been considerable recent work on approaches to programming normative systems. The JaCaMo programming framework combines the Jason [2], Cartago [13], and MOISE$^+$ [8] platforms. In this integrated approach, the organisational infrastructure of a multiagent system consists of organisational artefacts and

[3]This semantics is also used by Jason.

agents that together are responsible for the management and enactment of the organisation. An organisational artefact employs a normative program which in turn implements a MOISE$^+$ specification. A programming language for the implementation of normative programs as well as a translation of MOISE$^+$ specifications into normative programs is described in [9]. JaCaMo allows Jason agents to interact with organisational artefacts, e.g., to take on a certain role. (As the idea of organisational artefacts based on normative programs is closely related to the 2OPL architecture for normative systems [5] used in this paper, we believe that our N-2APL agents could also be used in the JaCaMo framework.) In contrast to N-2APL, the Jason agents in this combined model have no explicit mechanisms to reason about norms (obligations and prohibitions) and their deadlines and sanctions in order to adapt their behaviour at run time. Another approach that integrates norms in a BDI-based agent programming architecture is proposed in [11]. This extends the AgentSpeak(L) architecture with a mechanism that allows agents to behave in accordance with a set of non-conflicting norms. As in N-2APL, the agents can adopt obligations and prohibitions with deadlines, after which plans are selected to fulfil the obligations or existing plans are suppressed to avoid violating prohibitions. However, unlike N-2APL, [11] does not consider scheduling of plans with respect to their deadlines or possible sanctions.

There are also several agent languages which incorporate deadlines, including the Soft Real-Time Agent Architecture [16] and AgentSpeak(XL) [1]. These architectures use the TÆMS (Task Analysis, Environment Modelling, and Simulation) framework [6] together with Design-To-Criteria scheduling [17] to schedule intentions. TÆMS provides a high-level framework for specifying the expected quality, cost and duration of of methods (actions) and relationships between tasks (plans). Like N-2APL, tasks (and methods) can have deadlines, and TÆMS assumes that the expected execution times (and quality and costs) of tasks are available. As in N-2APL DTC can produce schedules which allow interleaved or parallel execution of tasks. However the view of 'deadline' used in these systems is different from that used here, in that deadlines are not hard (tasks still have value after their deadline), and they provide no support for normative concepts such as obligations, prohibitions and sanctions. To the best of our knowledge, they have not been used to develop norm-aware agents. AgentSpeak(RT) [15] is a version of AgentSpeak(L) which allows the specification of deadlines and priorities for tasks. However, as with SRTA and AgentSpeak(XL) it provides no support for normative concepts.

7. CONCLUSIONS

We have presented N-2APL, a programming language for norm-aware agents. N-2APL provides support for obligations with deadlines, prohibitions, and sanctions. N-2APL agents are guaranteed to be norm-aware rational, that is, to commit to a set of plans of the highest priority which do not violate higher priority prohibitions, and which are feasible. We believe that N-2APL represents a good compromise in the design of a norm-aware agent programming language. If the key assumptions underlying the design of N-2APL are relaxed, an agent either no longer satisfies norm-aware rationality, or its deliberation about goals, norms and sanctions is intractable. Although we have developed our ideas in the context of the 2APL agent programming language, the extensions to support norm-aware deliberation could be incorporated in a straightforward way to other BDI-based agent programming languages.

8. REFERENCES

[1] R. Bordini, A. L. C. Bazzan, R. d. O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In *Proc AAMAS'02*, pages 1294–1302, 2002.

[2] R. H. Bordini, M. Wooldridge, and J. F. Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[3] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[4] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.

[5] M. Dastani, D. Grossi, J.-J. C. Meyer, and N. Tinnemeier. Normative multi-agent programs and their logics. In *Proc. KRAMAS'09*, LNCS 5605, pages 16–31, 2009.

[6] K. S. Decker and V. R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 2:215–234, 1993.

[7] M. R. Garvey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 1979.

[8] J. Hübner, J. Sichman, and O. Boissier. $\mathcal{S} - \mathcal{M}$OISE$^+$: A middleware for developing organised multi-agent systems. In *Proc. COIN'06*, LNCS 3913, pages 64–78. Springer, 2006.

[9] J. F. Hübner, O. Boissier, and R. H. Bordini. From organisation specification to normative programming in multi-agent organisations. In Proc. CLIMA XI, LNCS 6245 , pages 117–134. Springer, 2010.

[10] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.

[11] F. R. Meneguzzi and M. Luck. Norm-based behaviour modification in BDI agents. In Proc. AAMAS'09, pages 177–184. IFAAMAS, 2009.

[12] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[13] A. Ricci, M. Viroli, and A. Omicini. Give agents their artifacts: the A&A approach for engineering working environments in MAS. In Proc. AAMAS'07, IFAAMAS, 2007.

[14] M. B. van Riemsdijk, K. V. Hindriks, and C. M. Jonker. Programming organization-aware agents: A research agenda. In *Proc. ESAW'09*, LNAI 5881, pages 98–112. Springer, 2009.

[15] K. Vikhorev, N. Alechina, and B. Logan. Agent programming with priorities and deadlines. In Proc. AAMAS'11, pages 397–404, 2011.

[16] R. Vincent, B. Horling, V. Lesser, and T. Wagner. Implementing soft real-time agent control. In *Proc. AGENTS'01*, pages 355–362, 2001.

[17] T. Wagner, A. Garvey, and V. Lesser. Criteria-directed heuristic task scheduling. *International Journal of Approximate Reasoning*, 19:91–118, 1998.