Run-Time Norm Compliance in BDI Agents

(Extended Abstract)

JeeHang Lee, Julian Padget
Department of Computer Science
University of Bath
Bath, BA2 7AY, UK
{j.lee, j.a.padget}@bath.ac.uk

Brian Logan, Daniela Dybalova, Natasha Alechina
School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK
{bsl, dxd, nza}@cs.nott.ac.uk

ABSTRACT

Normative systems offer a means to govern agent behaviour in dynamic open environments. Under the governance, individual agents themselves must reason about compliance with state- or event-based norms (or both) depending upon the formalism used. This paper describes how norm awareness enables a BDI agent to exhibit norm compliant behaviour at run-time taking into account normative factors. To this end, we propose ***N-Jason***, a run-time norm compliant BDI agent framework which supports norm-aware deliberation as well as a run-time norm execution mechanism, through which previously unknown norms are recognized and bring about the triggering of plans. To be able to process a norm such as an obligation, the agent architecture must be able to deal with deadlines and priorities, and choose between plans triggered by a particular norm. Consequently, we extend the syntax and the scheduling algorithm of AgentSpeak(RT) to operate in the context of ***Jason***/AgentSpeak(L) and provide 'real-time agency'.

Categories and Subject Descriptors

I.2.5 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Algorithms, Design, Language

Keywords

Norms, BDI, Agent Programming Language, Normative System

1. N-JASON: BDI AGENT FRAMEWORK

We propose ***N-Jason***, a BDI-based agent interpreter and a programming language for run-time norm compliance in agent behaviour. This extends ***Jason***/AgentSpeak(L) [2] in accordance with the 'real-time agency' of AgentSpeak(RT) [3], which supports normative concepts (i.e. obligations, permissions, prohibitions, deadlines, priorities and durations) enabling norm-aware deliberation. Consequently, run-time norm compliance is achieved in a single reasoning cycle by the interpreter through: (i) run-time norm execution, realised by event- and option- reconsideration at a perception stage (in a belief-update process more exactly), and (ii) norm-aware deliberation, accomplished by intention scheduling with deadlines and priorities in a practical reasoning process. Scheduled intentions are executed afterwards by the ***N-Jason*** agent.

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*

1.1 N-Jason Language Extensions

N-Jason agent consists of four main components: beliefs, goals, events and a set of plans. Beliefs and goals are identical to those in ***Jason*** (details can be found in [2]), while event and plan syntax is extended with *deadline*, *priority* and *duration*, in order to support normative concepts. A *deadline* is a real time value expressed in a some adequate unit or real world time. A *priority* is a positive integer value indicating a relative importance between achieving a goal and responding to belief changes. Both can be stipulated optionally in the annotation of events, such as *+!event[deadline(d), priority(p)]*. A *duration* is a non-negative integer value representing a required time to execute the plan. This also can be optionally specified in the annotation of a plan label, such as *@plan[duration(te)] +!event <- plan_body.*.

1.2 Run-Time Norm Execution

Our approach to run-time norm execution is to use the *"pre-existing capabilities"* in an agent program when an agent encounters a previously unknown (event-based) norm. This is carried out in two steps: (i) event reconsideration and (ii) option reconsideration. These reconsiderations are determined by the executability of the new and unknown norms. We say that a norm such as *obl(evt, deadline, violation)*, is executable at run-time iff:

1. $p \in P$ and $type(p) = (obligation \mid prohibition)$, where p is a percept, formed from a list of terms in a set of newly observed percepts P at run-time;
2. $te_p \notin E$, where te_p is a triggering event generated from the percept p, and E is an event base, a set of intentions $\{(te, \tau), (te', \tau'), \ldots\}$, of which event is a pair of a triggering event and an intention (te, τ);
3. $edp(p) \neq \emptyset$ and $\{te_{edp(p)}\} \cap E \neq \phi$, where $edp(p)$ is a function extracting the obliged event together with its deadline and priority from p and $te_{edp(p)}$ is a triggering event of the $edp(p)$, an event term in the norm, and
4. $R_{te_{edp(p)}} \neq \phi$, where $R_{te_{edp(p)}}$ is a relevant plans selection function.

Event Reconsideration aims to verify that a norm perceived at run-time is executable although no corresponding plan exists in the agent program. If an event extracted from a detached norm has a relevance to a certain set of plans, it thus has potential to trigger specific ones, and it is concluded that the norm is executable. In consequence, the interpreter adds the norm to the event base (E) as an achievement goal addition event. The procedure for event

Algorithm 1 Event Reconsideration

```
Require: $P := P \cup N$
Require: $te_p$ = create-tevent$(p)$
1: if $p \in P$ and $type(p) = obligation$ then
2:     $te_{edp(p)}$ = create-tevent$(edp(p))$
3:     $R_{te_{edp(p)}} := \{\pi\theta \mid \theta$ is a mgu for $te_{edp(p)}$ and plan $\pi\}$
4:     if $R_{te_{edp(p)}} \neq \phi$ then
5:         $E :=$add-event$(E, te_p)$
6:     end if
7: else if $p \in P$ and $type(p) = prohibition$ then
8:     $\Xi :=$add-prohibition$(\Xi, edp(p))$
9: end if
```

reconsideration is as follows (see Alg. 1): (1) to extract the terms representing an obliged event, a deadline and its priority[1] from the obligation using the function *edp* (line 2), (2) to construct a new triggering event, an achievement goal addition event in this case, from the combination of extracted terms (line 2), (3) to query the existence of a set of relevant plans to S_R with such a constructed triggering event (line 3) and (4) to add the triggering event to E, if relevant plans are retrieved (line 5). (5) if norm is a prohibition, then the extracted event is added into the prohibition base (Ξ) (line 7-8) and will be revisited at the norm deliberation stage.

Algorithm 2 Option Reconsideration

```
Require: $\langle te_p, \tau\rangle \in E$ where $te_p$ is an event and $\tau$ is an intention
Ensure: $\pi\theta\theta'$ where $\theta'$ is a context unifier for $te_{edp(p)}$ and plan $\pi$
1: if $type(p) = obligation$ then
2:     $te_{edp(p)}$ = create-tevent$(edp(p))$
3:     $R_{te_{edp(p)}} := \{\pi\theta \mid \theta$ is a mgu for $te_{edp(p)}$ and plan $\pi\}$
4:     if $R_{te_{edp(p)}} \neq \phi$ then
5:         $\pi\theta\theta' := S_o(te_{edp(p)})$ where $\theta'$ is a context unifier for
           $te_{edp(p)}$ and plan $\pi$
6:     end if
7: end if
```

Option-Reconsideration aims to determine an applicable plan corresponding to the new and unknown norm (whose executability is already verified) and is thus added into E as an achievement goal addition event. If the applicable plan is chosen, then it will probably be used to enact a norm-compliant behaviour, unless it is infeasible as judged by norm-aware deliberation. The procedure is shown in Alg. 2. At the beginning (line 1-3), the interpreter carries out exactly the same steps (1–3) as the event-reconsideration procedure. After that, the interpreter selects a single applicable plan as an intended means to which to commit (line 5).

1.3 Norm-Aware Deliberation

Norm awareness in the deliberation process is achieved by the scheduling of intentions with deadlines and priorities. We extend the algorithm proposed in [3] with the consideration of prohibitions in order to establish a conflict-free *preference maximal set* of intentions. The code is shown in Alg. 3. A set of candidate intentions $I_C = \{\tau, \tau', \dots\}$, which is sorted in descending order of a priority, is inserted into the scheduling process. If each intention is feasible, i.e. an intention can be executed before the deadline and is not prohibited by a set of prohibition $\Xi = \{\xi, \xi', \dots\}$, then the intention is added to the *preference maximal set* (Γ). Intentions that

[1]In principle, the last term is an event arising when a violation occurs. This value normally indicates the criticality of the violation. Higher values represents a higher priority.

Algorithm 3 Scheduling of Intentions

```
1: $\Gamma := \emptyset$
2: for all $\tau \in I$ in descending order of priority do
3:     if $\{\tau\} \cup \Gamma$ is feasible then
4:         if $(\tau \notin \Xi)$ or $(\tau \in \Xi$ and $\tau = \xi$ and $priority(\tau) >$
           $priority(\xi))$ then
5:             $\Gamma := \{\tau[p]\} \cup \Gamma$
6:         end if
7:     end if
8: end for
9: sort $\Gamma$ in order of increasing deadline
10: return $\Gamma$
```

cannot meet their deadline are dropped. The criteria are defined as follows:

1. An intention is feasible *iff* the execution of the intention is completed before its deadline, that is, for τ,

$$ne(\tau) + et(\tau) - ex(\tau) \leq dl(\tau)$$

   where τ denotes an intention, $ne(\tau)$ is the time at which τ will next execute, $et(\tau)$ is the time required to execute τ, denoted in the plan label, $ex(\tau)$ is the elapsed time to execute τ to this point, and $dl(\tau)$ is the deadline for τ specified in the plan [1].
2. The intention should not be prohibited, that is, for τ
   - $\tau \notin \Xi$ or
   - $\tau \in \Xi$, then $\exists\xi \in \Xi$, $\tau = \xi$ and $priority(\tau) > priority(\xi)$

   where τ is an intention, ξ is a prohibited event in the prohibition base Ξ and *priority* is a priority retrieval function.

2. CONCLUSION AND FUTURE WORK

We believe that a model for run-time norm compliance is beneficial for the enhancement of both norm compliance capability and agent autonomy from the agent's perspective, even though the behaviour generated by run-time norm execution may appear unexpected from the agent programmer's perspective. Although we only consider the execution of event-based norms at run-time, the extension to support state-based norms and its normative systems can easily be incorporated into ***N-Jason*** agents and will form part of future work. We also plan to detect violations which are generated in the norm aware deliberation, particularly when the normative goals are dropped during the scheduling. This offers a potentially useful link for enforcement in the context of normative system implementation.

3. REFERENCES

[1] N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 1057–1064, Richland, SC, 2012.

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

[3] K. Vikhorev, N. Alechina, and B. Logan. Agent programming with priorities and deadlines. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 397–404, Richland, SC, 2011.