Integrating Preferences in Reactive BDI Agents

Mostafa Mohajeri Parizi[1], **Giovanni Sileno**[2] and **Tom van Engers**[3]

Abstract.
The *belief-desire-intention* (BDI) model of agency is based on the notion of reactivity of agents towards internal or external events by selecting, instantiating and executing an applicable goal-plan rule from agent's procedural knowledge. Current BDI agent-architectures consider the reaction to an event by selecting plans via a "first applicable choice" principle. Recent works attempt to include in BDI agents explicit preferences, but do so as a "rationale" thus distorting their reactive nature and modifying the deliberation cycle. This paper proposes a method to include in the agent's specification preferences, and to use them in a manner that preserves reactivity.

keywords BDI agents, CP-nets, Preferences, Reactive agents

1 Introduction

Preferences are recognized to be a core concept of decision making [10]. This implies the general principle that when there are multiple goals that should be achieved or multiple ways to achieve a certain goal or even multiple sets of states that can be reached, the best course of action is the one that abides the most to agent's preferences [10]. An agent's preferences can vary from the implicit "maximize utility" in optimizing agent [9] to explicit preferences specified in a preference representation language [14].

Unexpectedly in all of the current BDI languages, preferences are only represented implicitly by the sequential ordering of plans provided by the modeler. There are works that add verbalized preferences to BDI agents [14, 9, 4], but these works all add preferences as a *rationale* step in the agent's deliberation cycle. This approach has a negative impact on the performance (because of the reflective step in the deliberation cycle) and the traceability of the script. We argue that modifying the deliberation cycle also affects the reactivity because if behavior is not (sufficiently) automatic the agent has to perform potentially complex, time-consuming deliberations to select the appropriate response to an event and for this she may become not (adequately) reactive. This insight is analytically supported a.o. by Heiner's theory of predictable behaviour [7]. To overcome these problems, this paper considers to integrate preferences as an explicit specification, to be used for an offline, pre-runtime step rewriting the agent script, and so guaranteeing both traceability and reactivity of the agent.

1.1 Background

Goal-plan rules BDI agents act based on their *goal-plan rules*, i.e. uninstantiated specifications of the means for achieving a goal [12]. These rules capture the procedural knowledge (*how-to*) of the agent. More formally, a goal-plan rule pr is a tuple $\langle e, c, p \rangle$, where: e is a *triggering event*, addressing a *invocation condition* or unistantiated goal; c, the *context condition*, is a first-order formula over the agent's belief base, making the rule *applicable*; p, the *plan body*, consisting of a finite sequence of steps $[a_1, a_2, ..., a_n]$ where each a_i is either a *goal* (i.e. an invocation attempting to trigger a goal-plan rule), or a *primitive action*. A goal-plan rule pr_i is then an *option* or a *possibility* for achieving a goal g, if the invocation condition of pr_i matches with g, and the preconditions of pr_i matches the current state of the world, as perceived or encoded in the agent's beliefs.

We will refer to a syntax close to that of AgentSpeak(L) [11]. As an example of script, consider:

```
+!g : c <= !a.
+!g <= !~b.
```

This code means that if the triggering event `+!g` occurs, if `c` holds, the agent commits to *achieve* `a`, otherwise (that is, `c` does not hold) the agent commits to *achieve* `~b`, or equivalently, to *escape* `b`.

Primitive actions Primitive actions are the lowest-level actions that can be used in the procedural knowledge of an agent; they are the actual means for the agent to change the environment (or itself) Also for primitve actions we will refer to the following syntax:

```
#a { c1 => +p,-q. c2 => +q. }
```

meaning that (the agent expects that) if the primitive action `a` occurs, if `c1` holds, then `p` will become true, if `c2` holds, then `q` will become true.

Preference Language In this work the explicit preferences of the agent are presented with CP-Nets. Conditional *ceteris paribus* preferences networks (CP-nets) are a compact representation of preferences in domains with finite *attributes of interest* [3]. An attribute of interest is an attribute in the world that the agent has some sort of preference over its possible values. CP-nets build upon the idea that most of the preferences people make explicit are expressed jointly with an implicit *ceteris paribus* ("all things being equal") assumption. For example, when people say "I prefer a sunny day to a rainy day", they do not mean at all costs and situations, but that they prefer sunny day, all other things being equal to their current situation. We denote the preference for achieving/maintaining the condition `q` over avoiding/escaping it in condition `c` as:

```
p > ~p : c.
```

2 Method

To integrate the CP-net preferences we need the contextualized outcome of each plan. To do so we translate the agent's script into an ASP program and use an off the shelf ASP solver to find out all the

[1] m.mohajeriparizi@uva.nl
[2] g.sileno@uva.nl
[3] tom.vanengers@tno.com

outcomes of each plan under each possible context condition, i.e. in each condition `c`, the most optimistic final state after execution of a plan is considered the outcome of adopting that plan in `c`. There are a few works in the literature that link BDI programs to logic programs [2], but for this work, we use an approach close to [6], in which a formal method for translation from HTN planning domain to logic programs is presented. The close connection between BDI and HTN has been explored extensively in the literature [5]. To support the incremental nature of scripts in a logic program we use *discrete event calculus* (DEC) [8]. We omit the full translation process in this abstract but an example translation can be found in [1].

We will present the preference integration method with an example: A player can play a match in three ways: just playing for fun, do whatever needed to win, or play conservative to avoid to lose. Suppose now that you might want e.g. to enjoy your game as much as winning, but you might also prefer to gain support from observers, unless your position in the ranking (captured by propositions as *first* and *last*) is really low. Let us start from the following (non-prioritized) procedural knowledge:

```
+!match => #funny_playing..
        => #robust_playing.
        => #opportunistic_playing.
```

And the following expectations about primitive actions:

```
#funny_playing { => +support, +enjoy, -win. }
#robust_playing { => +support, -lose, -enjoy. }
#opportunistic_playing { => +win, -support, -lose. }
```

We then specify our agent by the following preferences:

```
support > ~support : ~last.
enjoy > ~enjoy : first.
win > ~win : support, ~last.
win > ~win : last.
```

By translating this script to an ASP program and solving it with `clingo`, we obtain multiple answer sets. unique outcomes for all plans (in accordance to different context conditions). An example of condition and outcome extracted from a trace of plan `match[0]` is:

```
condition: ~win,~support,~enjoy,~lose,first,~last
outcome: ~win,support,enjoy,~lose,first,~last
```

Each answer set is evaluated in terms of the given CP-net preferences, resulting in a partial ordering between different outcomes of answer sets. For instance, the answer sets with outcome `1` are preferred over the answer sets with outcome `2`. The *dominance checking* is done with CRISNER tool [13].

```
outcome 1: ~win,support,enjoy,~lose,first,~last
outcome 2: win,~support,~enjoy,~lose,first,~last
```

Plan ranking Following the ranking, we can give a contextualized priority to plans. In each unique context condition `c`, a plan is preferred if it has a preferred outcome, observing that e.g. in the condition `~win,~support,~enjoy,~lose,first,~last` the plan `match[0]` is preferred to plan `match[1]` and in turn plan `match[1]` is preferred to `match[2]`.

Considering all existing plans, the initial procedural knowledge can be prioritized. For the 3 plans there are 6 possible orderings. The following code provides an example of conditional ordering obtained via our method, including a *boolean simplification* step for the preconditions:

```
+!match : (last | ~enjoy) & ~first
        <= !~lose.
        <= !win.
        <= !enjoy.
+!match : (~last | lose) & (last | enjoy) & ~win
        <= !enjoy.
        <= !win.
        <= !~lose.
```

In the rewritten script the plans are conditionally ordered in accordance to the explicit verbalized preferences provided by the designer.

3 Conclusion

This paper focuses on role of explicit preferences in reactive BDI scripts. A language based on CP-nets is used to add verbalized preferences to a BDI agent script and an off-line method is presented to *integrate* these preferences into the agent's procedural knowledge. The resulting script is explicitly prioritized and at the same time the reactivity of the agent is also maintained. An important aspect of this method is that it does not need any modification to the usual deliberation cycle of BDI frameworks and the resulting script can be run with any AgentSpeak(L)-like interpreter.

ACKNOWLEDGEMENTS

This paper results from work done within *Data Logistics for Logistics Data* project (DL4LD, `www.dl4ld.net`). The DL4LD is funded by the Dutch Science Foundation in the Commit2Data program (grant no: 628.001.001).

REFERENCES

[1] Example translation: https://gitlab.com/mohajeri/as2asp.
[2] Justin Blount, Michael Gelfond, and Marcello Balduccini, 'A theory of intentions for intelligent agents', *Lecture Notes in Computer Science*, **9345**, 134–142, (2015).
[3] Craig Boutilier, Ronen I Brafman, Carmel Domshlak, Holger H Hoos, and David Poole, 'CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *Journal of artificial intelligence research*, **21**, 135–191, (2004).
[4] Aniruddha Dasgupta and Aditya K. Ghose, 'Implementing reactive BDI agents with user-given constraints and objectives', *International Journal of Agent-Oriented Software Engineering*, **4**(2), 141, (2010).
[5] Lavindra de Silva, Lin Padgham, and Sebastian Sardina, 'HTN-like solutions for classical planning problems: An application to BDI agent systems', *Theoretical Computer Science*, **763**, 12–37, (2019).
[6] Jürgen Dix, Ugur Kuter, and Dana Nau, 'Planning in answer set programming using ordered task decomposition', in *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, volume 2821, pp. 490–504, (2003).
[7] By Ronald A Heiner, 'he Origin of Predictable Behavior Author (s): Ronald A . Heiner Source : The American Economic Review , Vol . 73 , No . 4 (Sep ., 1983), pp . 560-595 Published by : American Economic Association Stable URL : https://www.', **73**(4), 560–595, (1983).
[8] Erik T. Mueller, 'Event Calculus Reasoning Through Satisfiability', *J. Log. and Comput.*, **14**(5), 703–730, (October 2004).
[9] Ingrid Nunes and Michael Luck, 'Softgoal-based plan selection in model-driven BDI agents', *13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014*, **1**, 749–756, (2014).
[10] Gabriella Pigozzi, Alexis Tsoukiàs, and Paolo Viappiani, 'Preferences in artificial intelligence', *Annals of Mathematics and Artificial Intelligence*, **77**(3-4), 361–401, (2016).
[11] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Agents Breaking Away*, pp. 42–55, (1996).

[12] Anand S. Rao and Michael P. Georgeff, 'BDI agents: From theory to practice.', in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS1995)*, pp. 312–319, (1995).

[13] Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar, 'Dominance testing via model checking', in *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pp. 357–362, (2010).

[14] Simeon Visser, John Thangarajah, James Harland, and Frank Dignum, 'Preference-based reasoning in BDI agent systems', *Autonomous Agents and Multi-Agent Systems*, **30**(2), 291–330, (2016).