Reasoning About Preferences in BDI Agent Systems*

(Extended Abstract)

Simeon Visser
Utrecht University
Amsterdam, the Netherlands
simeon87@gmail.com

John Thangarajah
RMIT University
Melbourne, Australia
johnt@rmit.edu.au

James Harland
RMIT University
Melbourne, Australia
james.harland@rmit.edu.au

ABSTRACT

BDI agents often have to make decisions about which plan is used to achieve a goal, and in which order goals are to be achieved. In this paper we describe how to incorporate preferences (based on the $\mathcal{LPP}$ language) into the BDI execution model.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence — *Intelligent agents*

General Terms

Algorithms

Keywords

Agent programming languages, Reasoning (single and multiagent), Preference reasoning

1. INTRODUCTION

A fundamental feature of agent systems is the ability to make decisions and to manage the consequences of these decisions in complex dynamic environments. In agent systems based on the *Belief-Desire-Intention (BDI)* model, an agent typically has a set of *beliefs* about the current state of the world, a set of *goals*, which represent states of the world that it would like to bring about, and *plans* which are used to achieve its goals. Due to the unpredictability of an agent's environment, it is normal for the agent to have to choose one of several plans which may be used to achieve a particular goal; by suitably adapting the choice of plan for the circumstances applicable at the time, the agent can provide robust behavior.

For example, a travel agent that is asked to book a holiday may subdivide this task into two subgoals of booking accommodation and booking transport. If there are multiple accommodation venues and multiple means of transport, there can be numerous combinations that may be used by the agent to achieve the goal of booking a holiday.

In practice, it is common for the user to want to specify some preferences for how the goal should be achieved. For instance, in the travel example above, the user may wish to specify a particular choice of airline or that it is preferable to travel by train and spend any money saved on a better class of accommodation. This extra information should be included as a preference rather than a goal since, it is acceptable to satisfy the goal without satisfying the preference. For example, specifying the preference to fly with Dodgy Airlines as a goal would mean the user refuses to travel by any means other than Dodgy Airlines.

We have incorporated preferences into the BDI plan selection process by using preferences as a constraint on plan selection when a choice needs to be made. For example, if the user prefers 5^* hotels, then the agent should first choose plans which book 5^* hotels in preference to other plans. We also allow preferences to be specified for ordering subgoals of plans when their ordering is not determined by design. For example, satisfying the preference of travelling by train and spending any money saved on accommodation requires the subgoal of booking a train to be performed first.

2. PREFERENCE SPECIFICATION

Our preference specification consists of two parts: expressing the user preferences in a preference language and annotating the goals and plans of the agent with additional information. The annotated information is used at runtime when the agent utilizes the user preferences to make a decision.

Preferences are expressed in terms of *properties* of goals, which can be thought of as the relevant effects of the achievement of a goal. For example, a goal G of booking a holiday may have a property called *payment* which specifies the payment method used. Any plan that achieves G by paying for the holiday with a credit card will result in the value *credit* being assigned to this property. Similarly, an alternative plan may assign the value *debit* for *payment*. This means that the set $\{credit, debit\}$ contains the possible values of the property *payment* for G.

The intended meaning of a property p of a goal or plan is that upon successful execution of that goal or plan, the value of p will be either one of the programmer-specified values or a value called *null* when the agent's execution does not explicitly assign a value (e.g., a goal property may not receive a value if not all plans for that goal assign a value to that property).

Our preference language is based on the language $\mathcal{LPP}$ [1] and it allows the user to specify preferences over property values. For example, the statement "I would prefer for payment to be made via credit card" states the preference for the value *credit* rather than *debit* for the *payment* property.

The structure of our preference formulas follows $\mathcal{LPP}$ in

*We acknowledge the ARC Discovery Grant DP 1094627

Cite as: Reasoning About Preferences in BDI Agent Systems (Extended Abstract), Simeon Visser, John Thangarajah, James Harland, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1139-1140.

that we we use *basic desire formulas* to represent basic statements about the preferred situation, *atomic preference formulas* to represent an ordering over basic desire formulas and *general preference formulas* to express atomic preference formulas that are optionally subjected to a condition. We introduce the class of *conditional preference formulas* that allow us to specify conditions with regard to information collected at runtime. The user preferences are specified as a set of general preference formulas.

Due to space constraints we only give examples of each class of preference formulas and some user preferences together with their representation in our preference language. The semantics of our language is similar to that of $\mathcal{LPP}$ [1].

Examples of basic desire formulas are $transport.type = train$ and $usage(money, 500, \leq)$, indicating a preference for a preferred property value and the usage of a resource respectively. In atomic preference formulas we can order basic desire formulas to represent a preference of one over the other. For example, the atomic preference formula $transport.type = plane$ $(0) \gg transport.type = train$ (100) expresses that transport by plane is preferred to transport by train. A conditional preference formula, such as $failure(book\_flight)$, can be used to express preferences such as, "If I'm unable to travel by plane, then I prefer ..."

We now give several user preferences and their representation in our preference language. Examples of user preferences are "I prefer to minimize the money spent on accommodation.", "I prefer to fly rather than travel by train.", and "If the accommodation is a hotel then I prefer to fly with Jetstar.". We can represent the given user preferences as the following preference formulas:

$acc.minimize(money)$ (0)
$transport.type = plane$ $(0) \gg transport.type = train$ (100)
$acc.type = hotel : book\_flight.airline = Jetstar$ (0)

For the purpose of annotating and computing additional information for the goals and plans of the agent, we use the notion of a *goal-plan tree*. A goal-plan tree contains goal and plan nodes and it captures the decomposition of a goal into plans that can achieve that goal and the decomposition of a plan into subgoals that are posted by that plan. Specifically, in a goal-plan tree a goal node has one or more plan nodes as children and a plan node has zero or more goal nodes as children. We follow the approach of Thangarajah et al. [2, 3] to augment the nodes in a goal-plan tree with summary information. We annotate a node with a *property summary* containing properties with their possible values. We use resource summaries [3] to guide the agent's decisions with regard to preferences over resource usage.

For each goal node the programmer specifies a human-readable name and for each plan node the programmer can specify resource requirements and properties. For example, a goal named *book_hotel* can have a plan for booking a 3^* hotel (with resource requirement $money = 200$ and a property $quality = 3^*$) and a plan for booking a 5^* hotel (with $money = 400$ and $quality = 5^*$).

After annotating the goals and plans we propagate this information to nodes higher in the goal-plan tree. As a result, each property summary contains information of that node and all nodes below it in the goal-plan tree. We define two propagation rules that compute, for a given goal or plan node, the information in its property summary based on the annotations of that node and its child nodes. For example, the *book_hotel* goal above, assuming just the two plans mentioned as children, would have a resource summary of $\langle\ (money, 200), (money, 600)\ \rangle$[1] and a property summary of $\langle\ (quality, \{3^*, 5^*\})\ \rangle$[2] attached to its node in the tree.

We propagate information upwards to accumulate the available summary information in the root node (top-level goal) of the goal-plan tree. The user specifies preferences in terms of the summary information of the root node. The user therefore does not need to know the structure of the goal-plan tree. Further, the goal-plan tree can be used by multiple users as preferences are specified separately from it.

3. REASONING ABOUT PREFERENCES

We can identify two types of decisions that an agent needs to make. For a goal, an agent can select one of the plans and for a plan, an agent can choose the order in which to pursue the subgoals, if any, unless the order is determined by the structure of the plan.

The preferred order in which plans of a goal should be selected for execution is computed in two steps. We compute a score for each plan of a goal by evaluating the preference formulas and we then sort the plans by that score from most to least preferred. The output of this algorithm is an ordered list of the plans and the agent attempts the plans in that order. In case of plan failure, the next plan in the ordered list is attempted.

The order in which subgoals of a plan should be pursued is computed by analyzing the preference formulas containing a condition as well as the structure of the goal-plan tree. Consider the general preference formula

$$goal_1.prop_1 = value_1 : goal_2.prop_2 = value_2 \ (0)$$

which can be read as "if $prop_1$ of $goal_1$ has received the value $value_1$ then I prefer $prop_2$ of $goal_2$ to receive $value_2$". To satisfy this preference, we should execute $goal_1$ before $goal_2$ to determine the value of $prop_1$. If its value is indeed $value_1$ then we can aim to satisfy the preferred value of $prop_2$ for $goal_2$. We compute the constraints on subgoals for each plan (i.e. subgoal g_1 should preferably be executed before subgoal g_2) and we use these to compute the preferred order of subgoals of a plan. The execution order of subgoals of a plan is computed by repeatedly adjusting an ordering of the subgoals, starting with an arbitrary ordering, using the ordering constraints. For example, if g_1 should preferably be executed before g_2, we move g_2 to the end of the ordered list of subgoals and we proceed to the next ordering constraint.

We have implemented and tested our preference system in the agent platform Jadex[3] using a number of examples. The implementation consists of around 3000 lines of code, which utilizes the `metagoal` and `metaplan` features of Jadex.

4. REFERENCES

[1] M. Bienvenu, C. Fritz, and S. A. McIlraith. Planning with qualitative temporal preferences. In *KR*, pages 134–144. AAAI Press, 2006.

[2] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting & exploiting positive goal interaction in intelligent agents. In *AAMAS*, pages 401–408, 2003.

[3] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer. Avoiding resource conflicts in intelligent agents. In *ECAI*, pages 18–22, 2002.

[1]The necessary and possible resource requirements as described in [3].

[2]The property is assigned one and only of the values.

[3]`http://jadex.informatik.uni-hamburg.de`