Reasoning About Preferences in Intelligent Agent Systems *

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

Agent systems based on the BDI paradigm need to make decisions about which plans are used to achieve their goals. Usually the choice of which plan to use to achieve a particular goal is left up to the system to determine. In this paper we show how preferences, which can be set by the user of the system, can be incorporated into the BDI execution process and used to guide the choices made.

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 popular *Belief-Desire-Intention (BDI)* model [Rao and Georgeff, 1992], 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. The plans are often organized into a *plan library*, i.e. a collection of plans from which the agent makes a selection in order to achieve a particular goal. Due to the unpredictable nature of the environment in which agents are used, 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 use the number of alternatives in the plan library to provide robust behavior.

For example, a travel agent that is asked to book a holiday may adopt two subgoals of booking accommodation and booking transport. If there are 5 different accommodation venues and 6 different means of transport, there are a total of 30 different combinations that may be used by the agent to achieve the goal. However, not every such combination may be available; for instance, a given hotel or flight may be full. It is precisely this uncertainty that motivates the use of the BDI approach, in that we can specify what needs to be done whilst leaving the agent free to find the most appropriate combination of plans which will achieve this.

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 prefer a particular airline, or to travel by train and spend any money saved on a better class of accommodation. Note that this extra information is included as a preference rather than a goal, as it is acceptable to satisfy the goal without satisfying the preference. For example, if the user prefers to fly on Dodgy Airlines, but no such flights are available, then specifying this as a preference means that the user can still have a holiday; specifying this as a goal would mean that the user refuses to travel by any means other than Dodgy Airlines.

In this paper, we show how preferences can be incorporated into the BDI plan selection process, so that when a choice of plan is made, we can do so using the preferences as a constraint on plan selection. For example, if the user prefers 5^* accommodation, then the agent should first attempt to book accommodation of this type, i.e. choose plans which book 5^* hotels in preference to other plans. Similar comments apply to ordering subgoals in a plan, such as a preference for travelling by train and spending any money saved on accommodation requires means that the subgoal of booking the train be performed first.

We achieve this by adapting the preference language $\mathcal{LPP}$ [Baier and McIlraith, 2007; Bienvenu *et al.*, 2006] to specify preferences over *properties* of goals, which are a way of specifying what will occur when the goal is achieved. For example, booking a 5^* hotel gives the property *accommodation class* the value 5^*. The properties of a goal and their values are presented to the user, who can then use these to specify preferences, without having to know how the goals may be achieved. We use the notion of summary information [Thangarajah *et al.*, 2003; 2002] to derive the properties of a goal.

2 Preference Specification

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 possible values of the property $payment$ for goal G is the set $\{credit, debit\}$. The preference language we use al-

*We acknowledge the ARC Discovery Grant DP 1094627.

lows the user to specify preferences over the possible values of these properties. For example, the statement "I would prefer for payment to be made via credit" states the preference for value $credit$ rather than $debit$ for $payment$.

Preferences can also be specified considering the resource usage of plans used to achieve a goal. Preferences about resource usage is specified in terms of an amount and comparative operator for a particular resource type ("I prefer to spend at most $100 on transport").

The user then specifies preferences in terms of properties and resource usage of goals without having to know the details of how the goal is achieved. For example, for the above user, it is sufficient to know that paying for a holiday can be done by credit or debit without knowing that in order to pay, a mode of transport and accommodation must be selected.

The preferences we wish to express are concerned with the values of properties and the resource usage of goals. Examples include: "I prefer to minimize the money spent on accommodation.", "I prefer to fly rather than travel by train.", and "If the accommodation is a 5* hotel, I prefer to travel with Jetstar.". Our preference language is based on $\mathcal{LPP}$ [Baier and McIlraith, 2007; Bienvenu *et al.*, 2006]. Following the structure of $\mathcal{LPP}$, 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 preferences of a user are specified as a set of general preference formulas.*

Definition 2.1 (Basic Desire Formula) *A* basic desire formula *is either*

1. *the* $name$ *of a goal property and a desired or an undesired* $value$*, expressed as* $name = value$ *or* $name \neq value$*, where* $value$ *is not* $null$*,*[1]
2. *the predicate* minimize(resource)*, to express that the usage of* resource *should be minimized,*[2]
3. *the predicate* usage(resource, amount, comparator $\in \{<, \leq, =, \geq, >\}$) *to express that the usage of* resource *should be according to the* comparator *and the* amount *(e.g., at most 500), or*
4. *a predicate preceded by a goal name* goal *(e.g.* goal.minimize(resource)*).*

If $\varphi_1, \ldots, \varphi_n$*,* $n \geq 2$*, are of the form* $name = value$ *or* $name \neq value$*, then* $\varphi_1 \wedge \ldots \wedge \varphi_n$ *(conjunction) and* $\varphi_1 \vee \ldots \vee \varphi_n$ *(disjunction) are also basic desire formulas.*

Examples include $transport.type = train$ and $usage(money, 500, \leq)$. Basic desire formulas can be ordered in atomic preference formulas to express preferences in which a basic desire formula is preferred over another.

Definition 2.2 (Atomic Preference Formula) *Let* v_{min} *and* v_{max} *be numeric values such that* $v_{min} < v_{max}$*. An* atomic preference formula *is a formula*

$$\varphi_0\ (v_0) \gg \ldots \gg \varphi_n\ (v_n),\ n \geq 0,$$

where φ_i *is a basic desire formula and* $v_{min} \leq v_i \leq v_{max}$ *and* $v_i < v_j$ *for* $i < j$*. If* φ_i *is a predicate, then* φ_i *can only be used when* $i = n = 0$ *(i.e., we do not allow predicates together with other basic desire formulas).*

Note that we have dropped the constraint for φ_0 to have $v_0 = v_{min}$ as done by Bienvenu et al. and explain later the reason for this modification. We use v_{min}= 0 and v_{max}= 100 in our work. An example of an atomic preference formula is $transport.type = plane\ (0) \gg transport.type = train\ (100)$, which indicates that transport by plane is preferred over train. We now define conditional preference formulas which can be used in general preference formulas defined later.

Definition 2.3 (Conditional Preference Formula) *A* conditional preference formula *is*

1. *the* $name$ *of a goal property and a desired or an undesired* $value$*, expressed as* $name = value$ *or* $name \neq value$*, or*
2. *the predicate* success(goal)*,* failure(goal)*, or* used(goal, resource, amount)*, where* $goal$ *is the name of a goal,* $resource$ *is the name of a resource and* $amount \geq 0$ *is a numerical value.*

If $\varphi_1, \ldots, \varphi_n$*,* $n \geq 2$*, are conditional preference formulas then* $\varphi_1 \wedge \ldots \wedge \varphi_n$ *(conjunction) and* $\varphi_1 \vee \ldots \vee \varphi_n$ *(disjunction) are also conditional preference formulas.*

Some examples are $failure(book\_flight)$ and $success(transport)$. We now define general preference formulas to express atomic preference formulas that are optionally preceded by a conditional preference formula.

Definition 2.4 (General Preference Formula) *A* general preference formula *is*

1. *an atomic preference formula, or*
2. $\gamma : \Psi$*, where* γ *is a conditional preference formula and* Ψ *is an atomic preference formula.*

We can express the examples given in the beginning of this section as the following general preference formulas

$$accommodation.minimize(money)\ (0)$$
$$transport.type = plane\ (0) \gg transport.type = train\ (100)$$
$$accommodation.type = hotel\ \wedge\ accommodation.quality = 5* : book\_flight.airline = Jetstar\ (0)$$

Resource and property information are propagated upwards to the top level of the goal in a goal-plan tree. We follow a similar approach to the one taken by Thangarajah et al. [Thangarajah *et al.*, 2003; 2002] in which goals and plans are augmented with resource and effect summaries to detect and resolve resource conflicts and to facilitate the merging of common subgoals.

Typical BDI agent systems consist of a plan library where for a given goal, there is one or more plans in the plan library that could possibly achieve the goal (hence, this is an OR decomposition). Each plan performs some actions or posts a number of subgoals (this is an AND decomposition as all subgoals must be achieved). A subgoal is in turn handled by some other plan in the plan library. This decomposition leads

[1] We will explain later the purpose of $null$.

[2] Resource usage does not always need to be minimized when other preferences are taken into consideration.

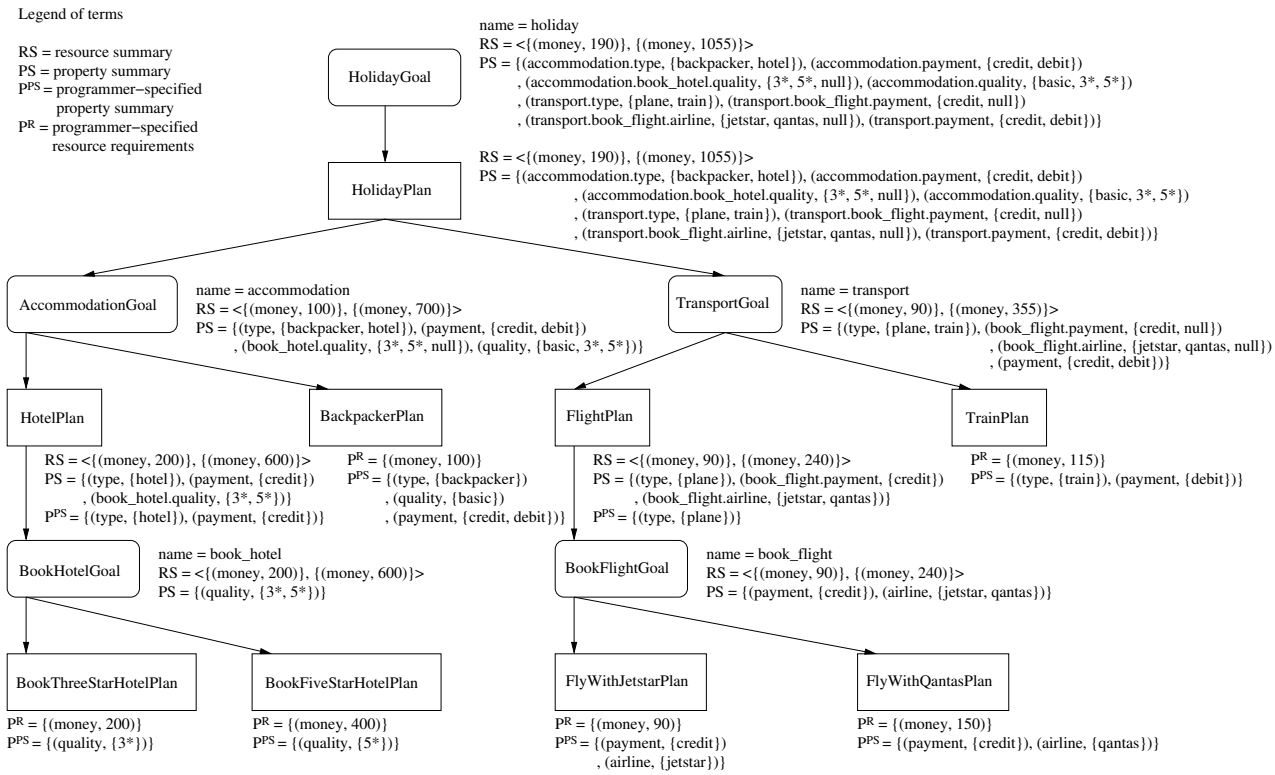

Figure 1: Goal-plan tree example

to a natural tree-like hierarchy which is termed 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. Figure 1 shows an example goal-plan tree of a goal of booking a holiday (HOLIDAYGOAL), which consists of two subgoals for booking accommodation (ACCOMMODATIONGOAL) and booking transport to the destination (TRANSPORTGOAL).

Similar to the approach of Thangarajah et al., we annotate nodes in the goal-plan tree with the required information and we automatically propagate this information to other nodes in the tree at compile time. The goal-plan tree in Figure 1 shows, apart from the goal-plan decompositions, both the programmer-specified properties P^{PS} and resources of plans P^R, and the propagated annotations of the summary information (property summaries PS and resource summaries RS).

We use resource summaries here as we also allow the user to specify preferences based on resources. A resource summary contains the *necessary* and *possible* resource usage for a node ($\langle N, P \rangle$). Necessary resources are those that are used irrespective of the goal-plan choice and possible resources are those that may be needed in the worst case where plans may fail and alternatives are tried. We define a *property summary* PS of a node N (denoted as PS_N) in the goal-plan tree, as a set of *properties*, where each property is of the form *(goalpath.name, values)* where *goalpath* denotes the path to N in the goal-plan tree, *name* is the name of the property, and *values* is a set of values. Note that $values$ is a non-empty set. The intended meaning of a property p in the property summary of a node N is that upon successful execution of N, the value of p is exactly one of that in $values$.

The human-readable names of the goals encountered from the root goal to the property's node (except the root node itself) are in $goalpath$. For example, the property summary of the node ACCOMMODATIONGOAL contains a property called $book\_hotel.quality$. This property has $book\_hotel$ as goalpath and $quality$ as the name.

We use the value $null \in values$ to indicate that the property could receive no value. For example, the goal TRANSPORTGOAL has two plans, FLIGHTPLAN and TRAINPLAN, where the former has the property $book\_flight.airline$ but the latter does not. Hence the property $book\_flight.airline$ of TRANSPORTGOAL will have the value $null$ if TRAINPLAN is successfully executed. Note that we allow a programmer-specified property to have more than one value. For example the property $payment$ of BACKPACKERPLAN has values $\{credit, debit\}$. If the user does not prefer a particular value, the resulting value of the property is irrelevant.

The nodes in the goal-plan tree are annotated with programmer-specified information and computed summary information. Each goal-node is annotated with the programmer defined $goalname$. For each plan in the plan library, we require that a programmer specifies the resources required by that plan ($P^{\mathcal{R}}$), similar to Thangarajah et al. [Thangarajah *et al.*, 2002], and additionally a property summary (P^{PS}). The corresponding plan-node in the goal-plan tree is annotated with this programmer-specified information.

For example, in Figure 1, goal BOOKFLIGHTGOAL is annotated with its name, and its plans FLYWITHJETSTARPLAN and FLYWITHQANTASPLAN are both annotated with the sets of resources and properties required by the plan. Note that it is sufficient to provide this information just for the lowest level plans (i.e. the leaves of the goal-plan tree) as the information is propagated up in the goal-plan tree. We do however allow annotation for all plan-nodes, such as HOTELPLAN which has both programmer-specified and computed information.

Using the programmer-specified information above, we compute the resource summaries using the techniques described by Thangarajah et al. [Thangarajah *et al.*, 2002], and the property summaries using similar principles. The techniques were shown to be computationally effective in [Thangarajah and Padgham, 2010]. Note that the summary information is computed at compile time and dynamically updated at run-time.

The goal-plan tree in Figure 1 illustrates the complete computed summaries for each node.

3 Reasoning about preferences

Consider the following preference formula

$book\_flight.airline \neq qantas : acc.type = hotel\ (0)$

Based on our previous observations, we want to execute BOOKFLIGHTGOAL before ACCOMMODATIONGOAL to obtain the value of $book\_flight.airline$. By analyzing the goal-plan tree, we see that BOOKFLIGHTGOAL is part of the subtree rooted at TRANSPORTGOAL. Further, both TRANSPORTGOAL and ACCOMMODATIONGOAL are subgoals of HOLIDAYPLAN. We see that at HOLIDAYPLAN, the agent should therefore pursue the subgoal TRANSPORTGOAL before ACCOMMODATIONGOAL.

Similarly, the preference $transport.type = plane\ (0) \gg transport.type = train\ (100)$ can be satisfied by choosing FLIGHTPLAN rather than TRAINPLAN to achieve TRANSPORTGOAL. We now describe algorithms that utilize user preferences and summary information in the goal-plan tree to guide these decisions.

When the agent needs to select a plan for a goal, our approach is to express numerically how well a plan satisfies the preference formulas. We can then sort the plans from most to least preferred and attempt the plans in that order to achieve the goal.

Before we can define the semantics of preference formulas, we first discuss the differences between preferences about goal properties and preferences about the resource usage of a goal. These preferences differ in the sense that goal properties and their possible values are more precisely known to the agent than the resource usage of a goal. The latter depends on the possibly unsuccessful execution of plans in the subtree rooted at that goal whereas the value of a goal property must be one of the values in the known set of values.

Since the actual resource usage of a node in the goal-plan tree is not known to the agent, and is dependent on the path chosen to achieve the goal, we have opted to use estimates of the amount that will be used. The resource summaries of a node only contain the amounts that will necessarily and possibly be used which is the only information the agent has with regard to the resource usage of a node. However, an estimate is necessary when attempting to satisfy preferences related to resource usage.

We use the function $k\texttt{-estimate}$ to compute the estimated resource usage of a particular resource of a plan.

Definition 3.1 (k-estimate) *Let (nec_i, pos_i) be the necessary and possible usage of a particular resource for a plan P_i. We define the* k-estimate *of P_i as $e_i = nec_i + k \cdot (pos_i - nec_i)$ where $0 \leq k \leq 1$.*

The value of k can be set for each plan and it is related to the expected failure rate of the plan, including the execution of its subgoals. For example, if a plan uses on average 10% more than the necessary amount of a resource, we can set $k = 0.1$. We emphasize that the estimate serves to guide the agent and that this does not mean the agent is able to execute the plan with a resource usage that is close to the estimate.

For example, the resource estimate for FLIGHTPLAN (see Figure 1), assuming k = 0.5, would be (90) + (0.5) . (240-90) = 165.

The semantics of our preference language are based on the semantics of $\mathcal{LPP}$. For each class of preference formulas, we define an evaluation function w that assigns a value $v_{min} \leq v \leq v_{max}$ to a formula of that class, where a lower value means more preferred. We evaluate preference formulas only for plan selection for a goal and we therefore evaluate a formula for a given goal G and a plan P_{eval} of G. We include G in the evaluation of a preference formula because the satisfaction of some basic desire formulas for a plan depends on the summary information of other plans of G. For example, the basic desire formula $minimize(resource)$ is satisfied for a plan if its resource usage of $resource$ is lowest compared to all other plans.

Compared to the work of Bienvenu et al., our main contributions are the semantics of our basic desire formulas and the semantics of our introduced class of conditional preference formulas. Otherwise, we follow their semantics, apart from the adaptation of general preference formulas to utilize conditional preference formulas.

Definition 3.2 (Basic Desire Satisfaction) *Let φ be a basic desire formula, let G be a goal and let $P_1, \ldots, P_n$, $n \geq 1$ be the plans for G. Furthermore, let PS_i and R_i respectively be the property summary and resource summary of P_i, and let PS_{eval} be the property summary of P_{eval}. We define $w(\varphi, G, P_{eval})$ as follows:*

- *if φ is $name = value$ then $w(\varphi, G, P_{eval}) = v_{min}$ iff there exists a property $p \in PS_{eval}$, such that $value \in values(p)$ and $name$ is equal to either*
  - *$name(p)$ or $name(p)$ prepended with the goalpath from the root node to G, excluding the name of the root goal node, or*
  - *$name(p)$ with the goalpath to G prepended, if there exists a plan $P_i \neq P_{eval}$ with a property $q \in PS_i$ such that q has no goalpath and $name(q) = name(p)$.*[3]

  Otherwise, $w(\varphi, G, P_{eval}) = v_{max}$.
- *if φ is $name \neq value$ then $w(\varphi, G, P_{eval})$ is as above for $(name = value)$ with the requirement that $value \notin values(p)$*
- *if φ is $\psi_1 \wedge \ldots \wedge \psi_m$, $m \geq 2$ then $w(\varphi, G, P_{eval}) = max(\{w(\psi_j, G, P_{eval}) \mid 0 \leq j \leq m\})$*

[3] This condition is used for properties that were merged using $\mathcal{N}$.

- *if* φ *is* $\psi_1 \vee \ldots \vee \psi_m$, $m \geq 2$ *then* $w(\varphi, G, P_{eval}) = min(\{w(\psi_j, G, P_{eval}) \mid 0 \leq j \leq m\})$
- *if* φ *is* $minimize(resource)$ *then* $w(\varphi, G, P_{eval}) = v_{min}$ *iff* $P_{eval} \in S$, *and* $w(\varphi, G, P_{eval}) = v_{max}$ *otherwise, where* S *is a set of plans computed by one of the following procedures.*[4]
  - `min_nec_pos` *Let* (nec_i, pos_i) *be the necessary and possible resource usage of* $resource$ *for* P_i. *Let* S *contain each plan* P_i *that satisfies*
    (1) there is no P_j, $i \neq j$ *such that* $nec_j < nec_i$,
    (2) if multiple plans satisfy (1), we include in S *only those for which it additionally holds that there is no* P_j, $i \neq j$ *such that* $pos_j < pos_i$.
  - `min_estimate` *Let* e_i *be the* k-`estimate` *of a plan* P_i. *Then* S *contains each plan* P_i *for which there is no* P_j, $i \neq j$ *such that* $e_j < e_i$.
- *if* φ *is* $usage(resource, amount, comparator)$ *then* $w(\varphi, G, P_{eval}) = v_{min}$ *iff* $P_{eval} \in S$, *and* $w(\varphi, G, P_{eval}) = v_{max}$ *otherwise, where* S *is a set of plans computed using the procedure corresponding to the* $comparator \in \{<, \leq, =, \geq, >\}$ *that is used.*
  - $<$ and $\leq$ $amount$: *Let* (nec_i, pos_i) *be the necessary and possible resource usage of* $resource$ *for* P_i. *Then* S *contains all plans* P_i *such that* $pos_i < amount$ *(resp.* $\leq$*). If no such plans exist, then* S *contains all plans* P_i *such that* $nec_i < amount$ *(resp.* $\leq$*).*
  - $= amount$: *Let* e_i *be the* k-`estimate` *for* P_i. *Let* S *contain all plans* P_i *such that* $|e_i - amount|$ *is lowest (i.e.,* e_i *is closest to* $amount$*).*
  - $>$ and $\geq$ $amount$: *Let* (nec_i, pos_i) *be the necessary and possible resource usage of* $resource$ *for* P_i. *Then* S *contains all plans* P_i *such that* $nec_i > amount$ *(resp.* $\geq$*). If no such plans exist, then* S *contains all plans* P_i *such that* $e_i > amount$ *(resp.* $\geq$*), where* e_i *is the* k-`estimate` *for* P_i.

As an illustration of the semantics of $name = value$, consider $accommodation.quality = 3^*$ when evaluating for the goal called $accommodation$. This formula is satisfied for HOTELPLAN because the property summaries of HOTELPLAN and BACKPACKERPLAN respectively contain a property called $book\_hotel.quality$ and $quality$ which means the second condition for $name$ is satisfied. Recall that an atomic preference formula orders one or more basic desire formulas, each with an associated value. The value we wish to associate with an atomic preference formula Φ as a whole is the value of a satisfied basic desire formula in Φ with the lowest associated value.

Definition 3.3 (Atomic Preference Satisfaction) *Let* $\Phi = \varphi_0\ (v_0) \gg \ldots \gg \varphi_n\ (v_n)$, $n \geq 0$, *be an atomic preference formula and let* G *be a goal. We define* $w(\Phi, G, P_{eval}) = v_i$ *if there exists* φ_i *such that* $w(\varphi_i, G, P_{eval}) = v_{min}$ *and there is no* $j < i$ *such that* $w(\varphi_j, G, P_{eval}) = v_{min}$. *Otherwise,* $w(\Phi, G, P_{eval}) = v_{max}$.

The evaluation of conditional preference formulas requires information about the execution of goals thus far. For this we define metadata $\mathcal{M} = \langle \mathcal{M}_P, \mathcal{M}_S, \mathcal{M}_F, \mathcal{M}_R \rangle$:

- $\mathcal{M}_P$ pairs of a goal name and its received value,
- $\mathcal{M}_S$ names of goals that have succeeded,
- $\mathcal{M}_F$ names of goals that have failed, and
- $\mathcal{M}_R$ a data structure with per goal and per resource the amount that was used. We use $\mathcal{M}_R^{goal}(resource)$ to denote the amount (possibly none) of $resource$ used thus far for the execution of $goal$.

Definition 3.4 (Conditional Preference Satisfaction) *Let* Φ *be a conditional preference formula and let metadata* $\mathcal{M} = \langle \mathcal{M}_P, \mathcal{M}_S, \mathcal{M}_F, \mathcal{M}_R \rangle$. *We define* $w(\Phi, \mathcal{M})$ *as follows*

- *if* Φ *is not a conjunction or disjunction, we define* $w(\Phi, \mathcal{M}) = v_{min}$ *iff the given condition holds and* $w(\Phi, \mathcal{M}) = v_{max}$ *otherwise.*
  - *if* Φ *is* $name = value$ *then* $(name, value) \in \mathcal{M}_P$
  - *if* Φ *is* $name \neq value$ *then* $(name, value) \notin \mathcal{M}_P$
  - *if* Φ *is* $success(goal)$ *then* $goal \in \mathcal{M}_S$
  - *if* Φ *is* $failure(goal)$ *then* $goal \in \mathcal{M}_F$
  - *if* Φ *is* $used(goal, resource, amount)$ *then* $\mathcal{M}_R^{goal}(resource) \geq amount$
- *if* Φ *is* $\varphi_i \wedge \ldots \wedge \varphi_n$, $n \geq 2$ *then* $w(\Phi, \mathcal{M}) = max(\{w(\varphi_i, \mathcal{M}) \mid 0 \leq i \leq n\})$
- *if* Φ *is* $\varphi_i \vee \ldots \vee \varphi_n$, $n \geq 2$ *then* $w(\Phi, \mathcal{M}) = min(\{w(\varphi_i, \mathcal{M}) \mid 0 \leq i \leq n\})$

Definition 3.5 (General Preference Satisfaction) *Let* Φ *be a general preference formula and let* G *be a goal. We define* $w(\Phi, G, P_{eval}, \mathcal{M})$ *as*

- $w(\varphi_0 \gg \ldots \gg \varphi_n, G, P_{eval}, \mathcal{M}) = w(\varphi_0 \gg \ldots \gg \varphi_n, G, P_{eval})$
- $w(\gamma : \Psi, G, P_{eval}, \mathcal{M}) = \begin{cases} v_{min}, & \text{if } w(\gamma, \mathcal{M}) = v_{max} \\ w(\Psi, G, P_{eval}), & \text{otherwise} \end{cases}$

Recall that in atomic preference formulas, we do not define φ_0 to have value $v_0 = v_{min}$ as done by Bienvenu et al. This allows preference formulas to override other formulas. For example, we can prefer $prop = val_1$ (100) and $\gamma : prop = val_2$ (50), where γ is a conditional preference formula. If γ is satisfied then a plan that satisfies $prop = val_2$ is preferred over a plan that satisfies $prop = val_1$.

We can now present our algorithm for computing the preferred order in which plans of a goal G should be selected for execution. The input of this algorithm is the set of general preference formulas $\mathcal{F}$, the goal G and metadata $\mathcal{M}$.

- For each plan P_i of G, compute a score $score_i$ which is the sum of the values of $w(f, G, P_i, \mathcal{M})$ for each $f \in \mathcal{F}$.
- Sort the plans by $score_i$ in non-decreasing order.

The output is an ordered list of the plans which the agent attempts in that order. In case of plan failure, the next plan in the ordered list is attempted.

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 wish to satisfy the user preferences as much as

[4]The procedure to be used is determined by the designer prior to execution.

possible which is why the agent should, given a general preference formula $\gamma : \Psi$, execute the goals mentioned in γ before the goals that are referred to in Ψ. We can determine these goal orderings at compile time by analyzing the preference formulas and the structure of the goal-plan tree.

4 Discussion

We have implemented our preference system in the agent platform Jadex[5] and have tested it on a number of examples, including the holiday example discussed above. The implementation consists of around 3000 lines of code, which utilizes the `metagoal` and `metaplan` features of Jadex. We extracted the goal-plan tree from the agent specification and we annotate and propagate the summary information before the system runs. We used the preference formulas from Section 2 and analysed the execution of our system both with and without preferences. When preferences were not used or when no preferences were available, the agent randomly selected a plan for a goal and pursued the subgoals of a plan in an arbitrary order. The user preferences were therefore only satisfied when the agent happened to make the right decisions during executions. When we included our reasoning algorithms, the agent booked backpacker accommodation (as cheapest alternative) and a flight with either airline as expected. In case the plan for backpacker accommodation failed, the agent booked a 3^* hotel. When this plan failed as well, the 5^* hotel was selected and due to the third preference formula, the agent selected the plan to fly with Jetstar rather than Qantas. Further, the accommodation goal was pursued before the transport goal to make this possible.

Preferences have been used for the selection [Hindriks *et al.*, 2008; Nguyen and Wobcke, 2006] and elimination [Hindriks and van Riemsdijk, 2008; Myers and Morley, 2001; 2002] of actions or plans. The preference language on which our work is based is also used by Fritz and McIlraith [Fritz and McIlraith, 2005; 2006] to integrate preferences into the agent programming language DT-Golog.

Various directions for future work exist. Firstly, we have not incorporated consistency checks to see if the user has specified preferences that conflict in some or all executions of the agent system. Secondly, we have not explored interactions that may exist between preferences: satisfying a preference in a part of the goal-plan tree might make it impossible to satisfy preferences in other parts of the goal-plan tree. Thirdly, we have focussed on consumable resources and we have not explored the specification of and reasoning about preferences over reusable resources. Lastly, the assumptions that we make in our propagation rules and the way the nodes in the goal-plan tree have been annotated influence the usefulness of the computed properties. For example in Figure 1, if BOOKFIVESTARHOTELPLAN was also annotated with $payment = \{credit\}$, we end up with $payment = \{credit\}$ and $book\_hotel.payment = \{credit, null\}$ for HOTELPLAN which is redundant. The underlying issue here is that, it is not clear whether a property $prop$ of a plan and a property $prop$ of a subgoal of that plan should be considered the same or not. Our future work will address the above limitations.

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

References

[Baier and McIlraith, 2007] Jorge A. Baier and Sheila A. McIlraith. On domain-independent heuristics for planning with qualitative preferences. In *7th Workshop on Nonmonotonic Reasoning, Action and Change (NRAC)*, 2007.

[Bienvenu *et al.*, 2006] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Planning with qualitative temporal preferences. In *KR*, pages 134–144. AAAI Press, 2006.

[Fritz and McIlraith, 2005] Christian Fritz and Sheila McIlraith. Compiling qualitative preferences into decision-theoretic golog programs. In *Proceedings of The 6th Workshop on Nonmonotonic Reasoning, Action, and Change*, Edinburgh, UK, 2005.

[Fritz and McIlraith, 2006] C. Fritz and S. McIlraith. Decision-theoretic golog with qualitative preferences. In *KR'2006*, pages 153–163, Lake District, UK, 2006.

[Hindriks and van Riemsdijk, 2008] K. Hindriks and B. van Riemsdijk. Using temporal logic to integrate goals and qualitative preferences into agent programming. In *DALT*, pages 215–232, 2008.

[Hindriks *et al.*, 2008] K. Hindriks, C. Jonker, and W. Pasman. Exploring heuristic action selection in agent programming. In *ProMAS*, volume 5442 of *Lecture Notes in Computer Science*, pages 24–39. Springer, 2008.

[Myers and Morley, 2001] K. L. Myers and D. N. Morley. Human directability of agents. In *K-CAP*, pages 108–115. ACM, 2001.

[Myers and Morley, 2002] K. L. Myers and D. N. Morley. Resolving conflicts in agent guidance. In *Proceedings of the Workshop on Preferences in AI and CP: Symbolic Approaches*, 2002.

[Nguyen and Wobcke, 2006] Anh Nguyen and Wayne Wobcke. An adaptive plan-based dialogue agent: integrating learning into a bdi architecture. In *AAMAS*, pages 786–788. ACM, 2006.

[Rao and Georgeff, 1992] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In *KR*, pages 439–449, 1992.

[Thangarajah and Padgham, 2010] John Thangarajah and Lin Padgham. Computationally effective reasoning about goal interactions. *Journal of Automated Reasoning*, pages 1–40, 2010.

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

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