TLP **6** (5): 559–607, 2006. © 2006 Cambridge University Press
doi:10.1017/S1471068406002717 Printed in the United Kingdom

*Planning with preferences using logic programming**

TRAN CAO SON and ENRICO PONTELLI

Knowledge Representation, Logic, and Advanced Programming Laboratory,
Computer Science Department, New Mexico State University, Las Cruces, NM, USA
(*e-mail:* {tson,epontell}@cs.nmsu.edu)

submitted 29 June 2004; revised 17 December 2004, 9 August 2005; accepted 29 August 2005

Abstract

We present a declarative language, $\mathcal{PP}$, for the high-level specification of preferences between possible solutions (or trajectories) of a planning problem. This novel language allows users to elegantly express non-trivial, multi-dimensional preferences and priorities over such preferences. The semantics of $\mathcal{PP}$ allows the identification of *most preferred trajectories* for a given goal. We also provide an answer set programming implementation of planning problems with $\mathcal{PP}$ preferences.

KEYWORDS: planning with preferences, preference language, preference representation, answer set planning

1 Introduction and motivation

Planning – in its classical sense – is the problem of finding a sequence of actions that achieves a predefined goal (Reiter 2001). Most of the research in AI planning has been focused on methodologies and issues related to the development of efficient planners. To date, several efficient planning systems have been developed (Long *et al.*). These developments can be attributed to the discovery of good domain-independent heuristics, the use of domain-specific knowledge, and the development of efficient data structures used in the implementation of planning algorithms. Logic programming has played a significant role in this line of research, providing a declarative framework for the encoding of different forms of knowledge and its effective use during the planning process (Son *et al.* 2005).

However, relatively limited effort has been placed on addressing several important aspects in real-world planning domains, such as *plan quality* and *preferences about plans*. In many real world frameworks, the space of feasible plans to achieve the goal is dense, but many of such plans, even if executable, may present undesirable

* This paper is an extended version of a paper that appeared in the *Proceedings of the 7th International Conference on Logic Programming and Non-Monotonic Reasoning*, 2004.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

features. In these frameworks, it may be simple to find a solution (*"a"* plan); rather, the challenge, is to produce a solution that is considered satisfactory w.r.t. the *needs* and *preferences* of the user. Thus, feasible plans may have a measure of quality, and only a subset of them may be considered acceptable. These issues can be seen in the following example.

Example 1
Let us consider planning problems in the *travel domain*. A planning problem in this domain can be represented by the following elements[1]:

- a set of fluents of the form $at(l)$, where l denotes a location, such as *home, school, neighbor, airport, etc.*;
- an initial location l_i;
- a final location l_f; and
- a set of actions of the form $method(l_1, l_2)$ where l_1 and l_2 are two distinct locations and *method* is one of the available transportation methods, such as *drive, walk, ride_train, bus, taxi, fly, bike, etc*. In addition, there might be conditions that restrict the applicability of actions in certain situations. For example, one can ride a taxi only if the taxi has been called, which can be done only if one has some money; one can fly from one place to another if he/she has the ticket; etc.

Problems in this domain are often rich in solutions because of the large number of actions which can be used in the construction of a plan. Consider, for example, a simple situation, in which a user wants to construct a 3-leg trip, that starts from a location l_1 and ends at l_4, and there are 10 ways to move along each leg, one of them being the action $walk(l_i, l_{i+1})$. The number of possible plans is 10^3 and

$$walk(l_1, l_2), walk(l_2, l_3), walk(l_3, l_4)$$

is a possible plan that achieves the goal. In most of the cases, the user is likely to dismiss this plan and to select another one for various reasons; among them the total distance from l_1 to l_4 might be too large, the time and/or energy required to complete the plan would be too much, etc. This plan, however, would be a reasonable one, and most likely the only acceptable solution, for someone wishing to visit his/her neighbors.

In selecting the plan deemed appropriate for him/herself, the user's preferences play an important role. For example, a car-sick person would prefer walking over driving whenever the action *walk* can be used. A wealthy millionaire cannot afford to waste too much time and would prefer to use a taxi. A poor student would prefer to bike over riding a taxi, simply because he cannot afford the taxi. Yet, the car-sick person will have to ride a taxi whenever other transportations are not available; the millionaire will have to walk whenever no taxi is available; and the student will have to use a taxi when he does not have time. In other words, there are instances where a user's preference might not be satisfied and he/she will have to use plans that do not satisfy such preference. □

[1] Precise formulae will be presented later.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

The above discussion shows that users' preferences play a decisive role in the choice of a plan. It also shows that hard-coding user preferences as a part of the goal is not a satisfactory way to deal with preferences. Thus, we need to be able to evaluate plan components at a finer granularity than simply as consistent or violated. In Myers and Lee (1999), it is argued that users' preferences are of vital importance in selecting a plan for execution when the planning problem has too many solutions. It is worth observing that, with a few exceptions – like the system SIPE-2 with meta-theoretic biases (Myers and Lee 1999) – most planning systems do not allow users to specify their preferences and use them in finding plans. The responsibility in selecting the most appropriate plan rests solely on the users. It is also important to observe that *preferences* are different from *goals* in a planning problem: a plan *must* satisfy the goal, while it may or may not satisfy the preferences. The distinction is analogous to the separation between *hard* and *soft* constraints (Bistarelli *et al.* 2000). For example, let us consider a user with the *goal* of being at the airport who *prefers* to use a taxi over driving his own car; considering his preference as a soft constraint, then the user will have to drive his car to the airport if no taxi is available; on the other hand, if the preference is considered as a hard constraint, no plan will achieves the user's goal when no taxi is available.

In this paper, we will investigate the problem of integrating user preferences into a planner. We will develop a *high-level language* for the specification of user preferences, and then provide a logic programming encoding of the language, based on Answer Set Programming (Niemelä 1999). As demonstrated in this work, normal logic programs with answer set semantics (Gelfond *et al.* 1990) provide a natural and elegant framework to effectively handle planning with preferences.

We divide the preferences that a user might have in different categories:

- *Preferences about a state:* the user prefers to be in a state s that satisfies a property ϕ rather than a state s' that does not satisfy it, in case both lead to the satisfaction of his/her goal;
- *Preferences about an action:* the user prefers to perform the action a, whenever it is feasible and it allows the goal to be achieved;
- *Preferences about a trajectory:* the user prefers a trajectory that satisfies a certain property ψ over those that do not satisfy this property;
- *Multi-dimensional Preferences:* the user has a *set* of preferences, with an ordering among them. A trajectory satisfying a more favorable preference is given priority over those that satisfy less favorable preferences.

It is important to observe the difference between ϕ and ψ in the above definitions. ϕ is a *state* property, whereas ψ is a formula over the whole *trajectory* (from the initial state to the state that satisfies the given goal).

The rest of this paper is organized as follows. In Section 2, we review the foundations of answer set planning. Section 3 presents the high-level preference language $\mathscr{PP}$. Section 4 describes a methodology to compute preferred trajectories using answer set planning. In Section 5 we discuss the related work, while Section 6 presents the final discussion and conclusions.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

2 Preliminary – answer set planning

In this section we review the basics of planning using logic programming with answer set semantics – *Answer Set Planning (or ASP)* (Dimopoulos *et al.* 1997; Lifschitz 2002; Subrahmanian and Zaniolo 1995). We will assume that the effect of actions on the world and the relationship between fluents in the world are expressed in an appropriate language. In this paper, we make use of the ontologies of a *variation* of the action description language $\mathscr{B}$ (Gelfond and Lifschitz 1998). In this language, an action theory is defined over two disjoint sets of names – the set of *actions* **A** and the set of *fluents* **F**. An action theory is a pair (D, I), where

- D is a set of propositions expressing the effects of actions, the relationship between fluents, and the executability conditions for the actions[2];
- I is a set of propositions representing the initial state of the world.

Instead of presenting a formal definition of $\mathscr{B}$, we introduce the syntax of the language by presenting an action theory representing the travel domain of Example 1. We write

- $at(l)$, where l is a constant representing a possible location, such as *home, airport, school, neighbor, bus_station*, to denote the fact that the agent[3] is at the location l;
- *available_car* to denote the fact that the car is available for the agent's use;
- $has_ticket(l_1, l_2)$ to denote the fact that the agent has the ticket to fly from l_1 to l_2; etc.

The action of driving from location l_1 to location l_2 causes the agent to be at the location l_2 and is represented in $\mathscr{B}$ by the following *dynamic causal law*:

$$drive(l_1, l_2) \textbf{ causes } at(l_2) \textbf{ if } at(l_1).$$

This action can only be executed if the car is available for the agent's use at the location l_1 and there is a road connecting l_1 and l_2. This information is represented by an *executability condition*:

$$drive(l_1, l_2) \textbf{ executable_if } available_car(l_1), at(l_1), road(l_1, l_2).$$

The fact that one can only be at one location at a time is represented by the following *static causal law* ($l_1 \neq l_2$):

$$\neg at(l_2) \textbf{ if } at(l_1).$$

Other actions with their executable conditions and effects are represented in a similar way.

[2] Executability conditions were not originally included in the definition of the language $\mathscr{B}$ in Gelfond and Lifschitz (1998).

[3] Throughout the paper, we assume that we are working in a single agent (or user) environment. Fluents and actions with variables are shorthand representing the set of their ground instantiations.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

To specify the fact that the agent is initially at home, he has some money, and a car is available for him to use, we write

initially $(at(home))$
initially (has_money)
initially $(available_car(home))$

Example 2
Below, we list some more actions with their effects and executability conditions, using $\mathscr{B}$.

$walk(l_1, l_2)$	**causes**	$at(l_2)$	**if**	$at(l_1), road(l_1, l_2)$
$bus(l_1, l_2)$	**causes**	$at(l_2)$	**if**	$at(l_1), road(l_1, l_2)$
$flight(l_1, l_2)$	**causes**	$at(l_2)$	**if**	$at(l_1), has_ticket(l_1, l_2)$
$take_taxi(l_1, l_2)$	**causes**	$at(l_2)$	**if**	$at(l_1), road(l_1, l_2)$
$buy_ticket(l_1, l_2)$	**causes**	$has_ticket(l_1, l_2)$		
$call_taxi(l)$	**causes**	$available_taxi(l)$	**if**	has_money
$rent_car(l)$	**causes**	$available_car(l)$	**if**	has_mony
$bus(l_1, l_2)$	**executable_if**	has_money		
$flight(l_1, l_2)$	**executable_if**	$connected(l_1, l_2)$		
$take_taxi(l_1, l_2)$	**executable_if**	$available_taxi(l_1)$		
$buy_ticket(l_1, l_2)$	**executable_if**	has_money		

where the l's denote locations, airports, or bus stations. The fluents and actions are self-explanatory. □

Since our main concern in this paper is not the language for representing actions and their effects, we omit here the detailed definition of the proposed variation of $\mathscr{B}$ (Gelfond and Lifschitz 1998). It suffices for us to remind the readers that the semantics of an action theory is given by the notion of *state* and by a *transition function* Φ, that specifies the result of the execution of an action a in a state s (denoted by $\Phi(a, s)$). Each state s is a set of fluent literals satisfying the two properties:

1. for every fluent $f \in \mathbf{F}$, either $f \in s$ or $\neg f \in s$ but $\{f, \neg f\} \nsubseteq s$; and
2. s satisfies the static causal laws.

A state s satisfies a fluent literal f (f holds in s), denoted by $s \models f$, if $f \in s$. A state s satisfies a static causal law

$$f \textbf{ if } p_1, \ldots, p_n$$

if, whenever $s \models p_i$ for every $1 \leqslant i \leqslant n$, then we have that $s \models f$. An action a is *executable* in a state s if there exists an executability condition

$$a \textbf{ executable_if } p_1, \ldots, p_n$$

in D such that $s \models p_i$ for every i, $\leqslant i \leqslant n$. An action theory (D, I) is *consistent* if

1. $s_0 = \{f \mid \textbf{initially } (f) \in I\}$ is a state, and
2. for every action a and state s such that a is executable in s, we have that $\Phi(a, s) \neq \emptyset$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

In this paper, we will assume that (D,I) is consistent. A *trajectory* of an action theory (D,I) is a sequence $s_0a_1s_1\ldots a_ns_n$ where s_i's are states, a_i's are actions, and $s_{i+1} \in \Phi(s_i, a_{i+1})$ for $i \in \{0,\ldots,n-1\}$.

A planning problem is specified by a triple $\langle D,I,G\rangle$, where (D,I) is an action theory and G is a fluent formula (a propositional formula constructed from fluent literals and propositional connectives) representing the goal. A possible solution to $\langle D,I,G\rangle$ is a trajectory $\alpha = s_0a_1s_1\ldots a_ms_m$, where $s_0 \models I$ and $s_m \models G$. In this case, we say that the trajectory α achieves G.

Answer set planning (Dimopoulos *et al.* 1997; Lifschitz 2002; Subrahmanian and Zaniolo 1995) solves a planning problem $\langle D,I,G\rangle$ by translating it into a logic program $\Pi(D,I,G)$ which consists of *(i)* rules describing D, I, and G; and *(ii)* rules generating action occurrences. It also has a parameter, *length*, declaring the maximal length of the trajectory that the user can accept. The two key predicates of $\Pi(D,I,G)$ are:

- $holds(f,t)$ – the fluent literal f holds at the time moment t; and
- $occ(a,t)$ – the action a occurs at the time moment t.

$holds(f,t)$ can be extended to define $holds(\phi,t)$ for an arbitrary fluent formula ϕ, which states that ϕ holds at the time moment t. Details about the program $\Pi(D,I,G)$ can be found in (Son *et al.* 2005)[4]. The key property of the translation of $\langle D,I,G\rangle$ into $\Pi(D,I,G)$ is that it ensures that each trajectory achieving G corresponds to an answer set of $\Pi(D,I,G)$, and each answer set of $\Pi(D,I,G)$ corresponds to a trajectory achieving G.

Theorem 1

(Son *et al.* 2005) For a planning problem $\langle D,I,G\rangle$ with a consistent action theory (D,I) and maximal plan length n,

1. if $s_0a_1\ldots a_ns_n$ is a trajectory achieving G, then there exists an answer set M of $\Pi(D,I,G)$ such that:
   (a) $occ(a_i,i-1) \in M$ for $i \in \{1,\ldots,n\}$, and
   (b) $s_i = \{f \mid holds(f,i) \in M\}$ for $i \in \{0,\ldots,n\}$.
2. if M is an answer set of $\Pi(D,I,G)$, then there exists an integer $0 \leqslant k \leqslant n$ such that $s_0a_1\ldots a_ks_k$ is a trajectory achieving G, where $occ(a_i,i-1) \in M$ for $1 \leqslant i \leqslant k$ and $s_i = \{f \mid holds(f,i) \in M\}$ for $i \in \{0,\ldots,k\}$.

In the rest of this work, if M is an answer set of $\Pi(D,I,G)$, then we will denote with α_M the trajectory achieving G represented by M. Answer sets of the program $\Pi(D,I,G)$ can be computed using answer set solvers such as **smodels** (Simons *et al.* 2002), **dlv** (Leone *et al.* 2005), **cmodels** (Lierler and Maratea 2004), **ASSAT** (Lin and Zhao 2002), and **jsmodels** (Le and Pontelli 2003).

[4] A Prolog program for translation $\langle D,I,G\rangle$ into $\Pi(D,I,G)$ can be found at `http://www.cs.nmsu.edu/~tson/ASPlan/Preferences/translate.pl`.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

3 A language for planning preferences specification

In this section, we introduce the language $\mathscr{PP}$ for planning preferences specification. This language allows users to express their preferences among plans that achieve the same goal. We subdivide preferences in different classes: *basic desires*, *atomic preferences*, and *general preferences*. Intuitively, a basic desire is a preference expressing a desirable property of a plan such as the use of certain action over the others, the satisfaction of a fluent formula, or a temporal property (Section 3.1). An atomic preference describes a one-dimensional ordering on plans and allows us to describe a ranking over the plans given a set of possibly conflicting preferences (Section 3.2). Finally, a general preference provides means for users to combine different preference dimensions (Section 3.3).

Let $\langle D, I, G\rangle$ be a planning problem with the set of actions **A** and the set of fluents **F**; let $\mathscr{F}_F$ be the set of all fluent formulae over **F**. The language $\mathscr{PP}$ is defined as special formulae over **A** and **F**. We will illustrate the different types of preferences using the action theory representing the travel domain discussed earlier (Example 2). User preferences about plans in this domain are often based on properties of actions. Some of these properties are flying is *very fast* but *very expensive*; walking is *slow*, and *very tiring* if the distance between the two locations is large but *cheap*; driving is *tiring* and *costs a little* but it is *cheaper* than flying and *faster* than walking; etc.

3.1 Basic desires

A basic desire is a formula expressing a single preference about a trajectory. Consider a user who is at *home* and wants to go to *school* (goal) spending as little money as possible (preference), i.e., his desire is to save money. He has only three alternatives: *walking, driving*, or *take_taxi*. Walking is the cheapest and riding a taxi is the most expensive. Thus, a preferred trajectory for him should contain the action *walk(.,.)*. This preference could also be expressed by a formula that forbids the fluent *available_taxi(home)* or *available_car(home)* to become true in every state of the trajectory, thus preventing him to drive or take a taxi to school. These two alternatives of preference representation are not always equivalent. The first one represents the desire of leaving a state using a specific group of actions, while the second one represents the desire of being in certain states.

Basic desires are constructed by using *state desires* and/or *goal preferences*. Intuitively, a state desire describes a basic user preference to be considered in the context of a specific state. A state desire φ (where φ is a fluent formula) implies that we prefer a state s such that $s \models \varphi$. A state desire $occ(a)$ implies that we prefer to leave state s using the action a. In many cases, it is also desirable to talk about the final state of the trajectory – we call this a *goal preference*. These cases are formally defined next.

Definition 1 (*State Desires and Goal Preferences*)
A (primitive) *state desire* is either a formula φ, where $\varphi \in \mathscr{F}_F$, or a formula of the form $occ(a)$, where $a \in \mathbf{A}$.
A *goal preference* is a formula of the form **goal**(φ), where φ is a formula in $\mathscr{F}_F$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

We are now ready to define a basic desire that expresses a user preference over the trajectory. As such, in addition to the propositional connectives $\wedge, \vee, \neg$, we will also use the temporal connectives **next**, **always**, **until**, and **eventually**.

Definition 2 (*Basic Desire Formula*)
A *basic desire formula* is a formula satisfying one of the following conditions:

- a goal preference φ is a basic desire formula;
- a state desire φ is a basic desire formula;
- given the basic desire formulae φ_1, φ_2, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\neg\varphi_1$, **next**(φ_1), **until**(φ_1, φ_2), **always**(φ_1), and **eventually**(φ) are also basic desire formulae.

For example, to express the fact that a user would like to take the taxi or the bus to go to school, we can write:

$$\textbf{eventually}(\ occ(bus(home, school)) \vee occ(taxi(home, school))\ ).$$

If the user's desire is not to call a taxi, we can write

$$\textbf{always}(\ \neg occ(call_taxi(home))\ ).$$

If for some reasons, the user's desire is not to see any taxi around his home, the desire

$$\textbf{always}(\ \neg available_taxi(home)\ ).$$

can be used. Note that these encodings have different consequences – the second prevents taxis to be present independently from whether it was called or not.

The definition above is used to develop formulae expressing a desire regarding the structure of trajectories. In the next definition, we will describe when a trajectory satisfies a basic desire formula. In a later section (Section 4), we will present logic programming rules that can be added to the program $\Pi(D, I, G)$ to compute trajectories that satisfy a basic desire. In the following definitions, given a trajectory $\alpha = s_0 a_1 s_1 \cdots a_n s_n$, the notation $\alpha[i]$ denotes the trajectory $s_i a_{i+1} s_{i+1} \cdots a_n s_n$.

Definition 3 (*Basic Desire Satisfaction*)
Let $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots a_n s_n$ be a trajectory, and let φ be a basic desire formula. α satisfies φ (written as $\alpha \models \varphi$) iff one of the following holds

- $\varphi =$ **goal**(ψ) and $s_n \models \psi$
- $\varphi = \psi \in \mathscr{F}_F$ and $s_0 \models \psi$
- $\varphi = occ(a)$, $(a_1 = a$ and $n \geqslant 1)$ or $(n = 0)$
- $\varphi = \psi_1 \wedge \psi_2$, $\alpha \models \psi_1$ and $\alpha \models \psi_2$
- $\varphi = \psi_1 \vee \psi_2$, $\alpha \models \psi_1$ or $\alpha \models \psi_2$
- $\varphi = \neg\psi$ and $\alpha \not\models \psi$
- $\varphi =$ **next**(ψ), $\alpha[1] \models \psi$, and $n \geqslant 1$
- $\varphi =$ **always**(ψ) and $\forall(0 \leqslant i \leqslant n)$ we have that $\alpha[i] \models \psi$
- $\varphi =$ **eventually**(ψ) and $\exists(0 \leqslant i \leqslant n)$ such that $\alpha[i] \models \psi$
- $\varphi =$ **until**(ψ_1, ψ_2) and $\exists(0 \leqslant i \leqslant n)$ such that $\alpha[j] \models \psi_1$ for all $0 \leqslant j < i$ and $\alpha[i] \models \psi_2$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Definition 3 allows us to check whether a trajectory satisfies a basic desire. This will also allow us to compare trajectories. Let us start with the simplest form of trajectory preference, involving a single desire.

Definition 4 (*Ordering Between Trajectories w.r.t. Single Desire*)
Let φ be a basic desire formula and let α and β be two trajectories. The trajectory α is *preferred* to the trajectory β (denoted as $\alpha \prec_\varphi \beta$) if $\alpha \models \varphi$ and $\beta \not\models \varphi$.

We say that α and β are *indistinguishable w.r.t.* φ (denoted as $\alpha \approx_\varphi \beta$) if one of the two following cases occur:

1. $\alpha \models \varphi$ and $\beta \models \varphi$, or
2. $\alpha \not\models \varphi$ and $\beta \not\models \varphi$.

Whenever it is clear from the context, we will omit φ from $\prec_\varphi$ and $\approx_\varphi$. We will also allow a weak form of single preference, described next.

Definition 5 (*Weak Single Desire Preference*)
Let φ be a basic desire formula and let α, β be two trajectories. α is *weakly preferred* to β (denoted $\alpha \preceq_\varphi \beta$) iff $\alpha \prec_\varphi \beta$ or $\alpha \approx_\varphi \beta$.

It is easy to see that $\approx_\varphi$ is an equivalence relation over the set of trajectories.

Proposition 1
Given a basic desire φ, the relation $\approx_\varphi$ is an equivalence relation.

Proof

1. *Reflexivity:* this case is obvious.
2. *Symmetry:* let us assume that, given two trajectories α, β we have that $\alpha \approx_\varphi \beta$. This implies that either both trajectories satisfy φ or neither of them do. Obviously, if $\alpha \models \varphi$ and $\beta \models \varphi$ ($\alpha \not\models \varphi$ and $\beta \not\models \varphi$) then we have also that $\beta \models \varphi$ and $\alpha \models \varphi$ ($\beta \not\models \varphi$ and $\alpha \not\models \varphi$), which leads to $\beta \approx_\varphi \alpha$.
3. *Transitivity:* let us assume that for the trajectories α, β, γ we have that

$$\alpha \approx_\varphi \beta \qquad \text{and} \qquad \beta \approx_\varphi \gamma$$

From the first component, we have two possible cases:

(a) $\alpha \models \varphi$ and $\beta \models \varphi$. Since $\beta \approx_\varphi \gamma$, we need to have $\gamma \models \varphi$, which leads to $\alpha \approx_\varphi \gamma$.
(b) $\alpha \not\models \varphi$ and $\beta \not\models \varphi$. This second component, together with $\beta \approx_\varphi \gamma$ leads to $\gamma \not\models \varphi$, and thus $\alpha \approx_\varphi \gamma$.

□

In the next proposition, we will show that $\preceq_\varphi$ is a partial order over the set of equivalence classes representatives of $\approx_\varphi$[5].

[5] This means that $\preceq_\varphi$ satisfies the following three properties: (i) Reflexivity: $\alpha \preceq_\varphi \alpha$; (ii) Antisymmetry: if $\alpha \preceq_\varphi \beta$ and $\beta \preceq_\varphi \alpha$ then $\alpha \approx_\varphi \beta$; and (iii) Transitivity: if $\alpha \preceq_\varphi \beta$ and $\beta \preceq_\varphi \gamma$ then $\alpha \preceq_\varphi \gamma$ where α, β, and γ are arbitrary trajectories.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Proposition 2
The relation $\preceq_\varphi$ defines a partial order over the set of representatives of the equivalence classes of $\approx_\varphi$.

Proof
Let us prove the three properties.

1. *Reflexivity:* consider a representative α. Since either $\alpha \models \varphi$ or $\alpha \not\models \varphi$, we have that $\alpha \preceq_\varphi \alpha$.
2. *Anti-symmetry:* consider two representatives α, β and let us assume that $\alpha \preceq_\varphi \beta$ and $\beta \preceq_\varphi \alpha$. Since both α and β are equivalent class representatives of $\approx_\varphi$, to prove this property, it suffices to show that $\alpha \approx_\varphi \beta$. First of all, we can observe that from $\alpha \preceq_\varphi \beta$ we have either $\alpha \prec_\varphi \beta$ or $\alpha \approx_\varphi \beta$. If $\alpha \prec_\varphi \beta$ then this means that $\alpha \models \varphi$ and $\beta \not\models \varphi$. But this would imply that $\beta \not\preceq_\varphi \alpha$. Then we must have that $\alpha \approx_\varphi \beta$.
3. *Transitivity:* consider three representatives $\alpha_1, \alpha_2, \alpha_3$ and let us assume

$$\alpha_1 \preceq_\varphi \alpha_2 \wedge \alpha_2 \preceq_\varphi \alpha_3$$

Let us consider two cases.

- $\alpha_1 \prec_\varphi \alpha_2$. This implies that $\alpha_1 \models \varphi$ and $\alpha_2 \not\models \varphi$. Because $\alpha_2 \preceq_\varphi \alpha_3$, we have that $\alpha_3 \not\models \varphi$. This, together with $\alpha_1 \models \varphi$, allows us to conclude $\alpha_1 \prec_\varphi \alpha_3$.
- $\alpha_1 \approx_\varphi \alpha_2$, then either $\alpha_1 \models \varphi$ and $\alpha_2 \models \varphi$, or $\alpha_1 \not\models \varphi$ and $\alpha_2 \not\models \varphi$. In the first case, we have $\alpha_1 \approx_\varphi \alpha_3$ if $\alpha_3 \models \varphi$ and $\alpha_1 \prec_\varphi \alpha_3$ if $\alpha_3 \not\models \varphi$, i.e., $\alpha_1 \preceq_\varphi \alpha_3$. If instead we have the second possibility, then since $\alpha_2 \not\models \varphi$ and $\alpha_2 \preceq_\varphi \alpha_3$, we must have $\alpha_3 \not\models \varphi$. This allows us to conclude that $\alpha_1 \approx_\varphi \alpha_3$ and thus $\alpha_1 \preceq_\varphi \alpha_3$.

□

We next define the notion of most preferred trajectories.

Definition 6 (*Most Preferred Trajectory w.r.t. Single Desire*)
Let φ be a basic desire formula. A trajectory α is said to be a *most preferred trajectory* w.r.t. φ if there is no trajectory β such that $\beta \prec_\varphi \alpha$.

Note that in the presence of preferences, a most preferred trajectory might require extra actions that would have been otherwise considered unnecessary.

Example 3
Let us enrich the action theory of Example 2 with an action called *buy_coffee*, which allows one to have coffee, i.e, the fluent *has_coffee* becomes true. The coffee is not free, i.e., the agent will have to pay some money if he buys coffee. This action can only be executed at the nearby Starbucks shop. If our agent wants to be at school and prefers to have coffee, we write:

$$\textbf{goal}(\mathit{has_coffee}).$$

Any plan satisfying this preference requires the agent to stop at the Starbucks shop before going to school. For example, while $s_0 walk(\mathit{home}, \mathit{school}) s_1$, where s_0 and s_1

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

denote the initial state (the agent is at home) and the final state (the agent is at school), respectively, is a valid trajectory for the agent to achieve his goal, this is not a most preferred trajectory; instead, the agent has to go to the Starbucks shop, buy the coffee, and then go to school. Besides the action of *buy_coffee* that is needed for him to get the coffee, the most preferred trajectory requires the action of *going to the coffee shop*, which is not necessary if he does not have the preference of having the coffee.

Observe that the most preferred trajectory contains the action *buy_coffee*, which can only be executed when the agent has some money. As such, if the agent does not have any money, this action will not be executable and no trajectory achieving the goal of being at the school will contain this action. This means that no plan can satisfy the agent's preference, i.e., he will have to go to school without coffee. □

The above definitions are also expressive enough to describe a significant portion of preferences that frequently occur in real-world domains. Since some of them are particularly important, we will introduce some syntactic sugar to simplify their use:

- (Strong Desire) given the basic desire formulae φ_1, φ_2, $\varphi_1 < \varphi_2$ denotes $\varphi_1 \wedge \neg\varphi_2$.
- (Weak Desire) given the basic desire formulae φ_1, φ_2, $\varphi_1 <^w \varphi_2$ denotes $\varphi_1 \vee \neg\varphi_2$.
- (Enabled Desire) given two actions a_1, a_2, we will denote with $a_1 <^e a_2$ the formula $(executable(a_1) \wedge executable(a_2)) \Rightarrow (occ(a_1) < occ(a_2))$ where

$$executable(a) = \bigvee_{a \textbf{ executable_if } p_1,\ldots,p_k} p_1 \wedge \ldots \wedge p_k.$$

In the rest of the paper, we often use the following shorthands:

- For a sequence of preference formulae $\varphi_1, \ldots, \varphi_k$,

$$\varphi_1 < \ldots < \varphi_k$$

stands for

$$\bigwedge_{i \in \{1,\ldots,k-1\}} (\varphi_i < \varphi_{i+1}).$$

- For a sequence of preference formulae $\varphi_1, \ldots, \varphi_k$,

$$\varphi_1 <^w \ldots <^w \varphi_k$$

stands for

$$\bigwedge_{i \in \{1,\ldots,k-1\}} (\varphi_i <^w \varphi_{i+1}).$$

- For the sequence of actions $a_1, \ldots, a_k, b_1, \ldots, b_m$,

$$(a_1 \vee \ldots \vee a_k) <^e (b_1 \vee \ldots \vee b_m)$$

is a shorthand for

$$\bigwedge_{i \in \{1,\ldots,k\},\, j \in \{1,\ldots,m\}} (a_i <^e b_j).$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- For actions with parameters like *drive* or *walk*, we sometime write $drive <^e walk$ to denote the preference

$$\bigvee_{l_1,l_2 \in S,\, l_1 \neq l_2} (drive(l_1, l_2) <^e walk(l_1, l_2)).$$

where S is a set of pre-defined locations. Intuitively, this preference states that we prefer to drive rather than to walk between locations belonging to the set S. For example, if $S = \{home, school\}$ then this preference says that we prefer to drive from home to school and vice versa.

We can prove the following simple property of $\leqslant^w$.

Lemma 1
Consider the set of basic desire formulae and let us interpret $<^w$ as a relation. This relation is transitive.

Proof
Let $\varphi_1 <^w \varphi_2$ and $\varphi_2 <^w \varphi_3$. But these are the same as

$$(\varphi_1 \vee \neg\varphi_2) \wedge (\varphi_2 \vee \neg\varphi_3)$$

which implies

$$\varphi_1 \vee \neg\varphi_3$$

and thus $\varphi_1 <^w \varphi_3$. □

3.2 Atomic preferences

Basic desires allow the users to specify their preferences and can be used in selecting trajectories which satisfy them. From the definition of a basic desire formula, we can assume that users always have a set of desire formulae and that their intention is to find a trajectory that satisfies all such formulae. In many cases, this proves to be too strong and results in situations where no preferred trajectory can be found. Consider again the preference in Example 3, it is obvious that the user cannot have a plan that costs him nothing and yet satisfies his preferences. In the travel domain, *time* and *cost* are two criteria that a user might have when making a travel plan. These two criteria are often in conflict, i.e., a transportation method that takes little time often costs more. As such, it is very unlikely that the user can get a plan that costs very little and takes very little time.

Example 4
Let us continue with our travel domain. Again, let us assume that the agent is at home and he wants to go to school. To simplify the representation, we will write *bus*, *taxi*, *drive*, and *walk* to say that the agent takes the bus, taxi, drive, or walk to school, respectively. The following is a desire expressing that the agent prefers to get the fastest possible way to go to school (assume that both *driving* and *taking the bus* require about the same amount of time to reach the school):

$$time = \textbf{always}(taxi <^e (drive \vee bus) <^e walk)$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

On the other hand, when the agent is not in a hurry, he/she prefers to get the cheaper way to go to school (assume that *driving* and *taking the bus* cost about the same amount of money):

$$cost = \textbf{always}(walk <^{e} (drive \vee bus) <^{e} taxi)$$

Here, the preference states that the agent prefers to execute first the action that consumes the least amount of money. □

It is easy to see that any trajectories satisfying the preference *time* will not satisfy the preference *cost* and vice versa. This discussion shows that it is necessary to provide users with a simple way to *rank* their basic desires. To address the problem, we allow a new type of formulae, called *atomic preferences*, which represents an ordering between basic desire formulae.

Definition 7 (*Atomic Preference*)
An *atomic preference formula* is defined as a formula of the type $\varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_n$ where $\varphi_1, \ldots, \varphi_n$ are basic desire formulae.

The intuition behind an atomic preference is to provide an ordering between different desires – i.e., it indicates that trajectories that satisfy φ_1 are preferable to those that satisfy φ_2, etc. Clearly, basic desire formulae are special cases of atomic preferences – where all preference formulae have the same rank. The definitions of $\approx$ and $\prec$ can now be extended to compare trajectories w.r.t. atomic preferences.

Definition 8 (*Ordering Between Trajectories w.r.t. Atomic Preferences*)
Let α, β be two trajectories, and let $\Psi = \varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_n$ be an atomic preference formula.

- α, β are *indistinguishable* w.r.t. Ψ (written as $\alpha \approx_{\Psi} \beta$) if

$$\forall i.\, (\, 1 \leqslant i \leqslant n \Rightarrow \alpha \approx_{\varphi_i} \beta\,)\,.$$

- α is *preferred* to β w.r.t. Ψ (written as $\alpha \prec_{\Psi} \beta$) if $\exists (1 \leqslant i \leqslant n)$ such that
  1. $\forall (1 \leqslant j < i)$ we have that $\alpha \approx_{\varphi_j} \beta$, and
  2. $\alpha \prec_{\varphi_i} \beta$.

We will say that $\alpha \preceq_{\Psi} \beta$ if either $\alpha \prec_{\Psi} \beta$ or $\alpha \approx_{\Psi} \beta$.

It is easy to see that $\approx_{\Psi}$ is an equivalence relation on the set of trajectories. The following proposition is similar to Proposition 2.

Proposition 3
For an atomic preference Ψ, $\preceq_{\Psi}$ is a partial order over the set of representatives of the equivalence classes of $\approx_{\Psi}$.

Proof
Let us analyze the three properties.

- *Reflexivity:* Consider a representative α. By Definition 8, $\alpha \approx_{\Psi} \alpha$, which leads to $\alpha \preceq_{\Psi} \alpha$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- *Anti-symmetry:* Let $\alpha \preceq_\Psi \beta$ and $\beta \preceq_\Psi \alpha$. Again, it is enough if we can show that $\alpha \approx_\Psi \beta$. Let us assume, by contradiction, that $\alpha \prec_\Psi \beta$. This means that there is a value of i such that, for all $1 \leqslant j < i$ we have that $\alpha \approx_{\varphi_j} \beta$ and $\alpha \prec_{\varphi_i} \beta$. But this implies that $\beta \approx_{\varphi_j} \alpha$ for $j < i$ and $\beta \not\prec_{\varphi_i} \alpha$, which ultimately means $\beta \not\preceq_\Psi \alpha$, contradicting the initial assumptions.
- *Transitivity:* let $\alpha_1, \alpha_2, \alpha_3$ be three representatives such that

$$\alpha_1 \preceq_\Psi \alpha_2 \wedge \alpha_2 \preceq_\Psi \alpha_3$$

Let us consider the possible cases arising from the first component:

— $\alpha_1 \approx_\Psi \alpha_2$. This means that $\alpha_1 \approx_{\varphi_j} \alpha_2$ for all $1 \leqslant j \leqslant n$. We have two sub-cases:

  – $\alpha_2 \approx_\Psi \alpha_3$. Because $\approx_\Psi$ is an equivalence relation, we have that $\alpha_1 \approx_\Psi \alpha_3$, which implies that $\alpha_1 \preceq_\Psi \alpha_3$.

  – $\alpha_2 \prec_\Psi \alpha_3$. This means that there exists i, $1 \leqslant i \leqslant n$, such that $\alpha_2 \approx_{\varphi_k} \alpha_3$ for all $1 \leqslant k < i$ and $\alpha_2 \prec_{\varphi_i} \alpha_3$. Since $\approx_{\varphi_j}$ is an equivalence relation, we have that $\alpha_1 \approx_{\varphi_j} \alpha_3$ for all $1 \leqslant j < i$. Furthermore, $\alpha_1 \approx_{\varphi_i} \alpha_2$ and $\alpha_2 \prec_{\varphi_i} \alpha_3$ imply that $\alpha_1 \models \varphi_i$, $\alpha_2 \models \varphi_i$, and $\alpha_3 \not\models \varphi_i$. Thus, $\alpha_1 \prec_{\varphi_i} \alpha_3$. Hence, $\alpha_1 \preceq_\Psi \alpha_3$.

— $\alpha_1 \prec_\Psi \alpha_2$. This implies that there exists i, $1 \leqslant i \leqslant n$, such that $\alpha_1 \approx_{\varphi_k} \alpha_2$ for all $1 \leqslant k < i$ and $\alpha_1 \prec_{\varphi_i} \alpha_2$. Again, we have two sub-cases:

  – $\alpha_2 \approx_\Psi \alpha_3$. This means that $\alpha_2 \approx_{\varphi_j} \alpha_3$ for all j, $1 \leqslant j \leqslant n$. So, we have that $\alpha_1 \approx_{\varphi_j} \alpha_3$ for all $1 \leqslant j < i$, since $\approx_{\varphi_j}$ is an equivalence relation. Similar to the above case, we can show that $\alpha_1 \prec_{\varphi_i} \alpha_2$ and $\alpha_2 \approx_{\varphi_i} \alpha_3$ implies that $\alpha_1 \prec_{\varphi_i} \alpha_3$. Thus, $\alpha_1 \prec_\Psi \alpha_3$.

  – $\alpha_2 \prec_\Psi \alpha_3$. This means that there exists j, $1 \leqslant j \leqslant n$, such that $\alpha_2 \approx_{\varphi_k} \alpha_3$ for all $1 \leqslant k < j$ and $\alpha_2 \prec_{\varphi_j} \alpha_3$. If $j \geqslant i$, we have that $\alpha_1 \approx_{\varphi_k} \alpha_3$ for $k < i$ and $\alpha_1 \prec_{\varphi_i} \alpha_3$ (because $\alpha_1 \prec_{\varphi_i} \alpha_2$ and $\alpha_2 \approx_{\varphi_i} \alpha_3$). Otherwise, if $j < i$, using this fact and the transitivity of $\approx_{\varphi_k}$, we can conclude that $\alpha_1 \approx_{\varphi_k} \alpha_3$ for all $1 \leqslant k < j$ and $\alpha_1 \prec_{\varphi_j} \alpha_3$, which implies that $\alpha_1 \prec_\Psi \alpha_3$.

□

Definition 9 (*Most Preferred Trajectory w.r.t. Atomic Preferences*)
A trajectory α is *most preferred* if there is no other trajectory that is preferred to α.

Example 5
Let us continue the Example 4. The two preferences *time* and *cost* can be combined into different atomic preferences, e.g.,

$$time \lhd cost \qquad \text{or} \qquad cost \lhd time.$$

The first one is more appropriate for the agent when he is in a hurry while the second one is more appropriate for him when he has time. The trajectory

$$\alpha = s_0\ walk(home, school)\ s_1$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

is preferred to the trajectory

$$\beta = s_0\ call_taxi(home)\ s_1'\ taxi(home, school)\ s_2'$$

w.r.t. to the preference $cost \lhd time$, i.e., $\alpha \prec_{cost \lhd time} \beta$.[6] On the other hand, we have that $\beta \prec_{time \lhd cost} \alpha$. □

3.3 General preferences

Atomic preferences allow users to list their preferences according to their importance, where more preferred desires appear before less preferred ones. Naturally, when a user has a set of atomic preferences, there is a need for combining them to create a new preference that can be used to select the best possible trajectory suitable to him/her. This can be seen in the next example.

Example 6
Let us continue with the action theory described in Example 1. Besides *time* and *cost*, agents often have their preferences based on the level of comfort and/or safety of the the available transportation methods. This preferences can be represented by the formulae

$$comfort = \textbf{always}(flight <^e (drive \vee bus) <^e walk)$$

and

$$safety = \textbf{always}(walk <^e flight <^e (drive \vee bus)).$$

Now, consider an agent who has in mind the four basic desires *time, cost, comfort,* and *safety*. He can rank these preferences and create different atomic preferences, i.e., different orders among these preferences. Let us assume that he has combined these four desires and produced the following two atomic preferences

$$\Psi_1 = comfort \lhd safety \quad \text{and} \quad \Psi_2 = cost \lhd time.$$

Intuitively, Ψ_1 is a comparison between level of comfort and safety, while Ψ_2 is a comparison between affordability and duration.

Suppose that the agent would like to combine Ψ_1 and Ψ_2 to create a preference stating that he prefers trajectories that are as comfortable as possible and cost as little as possible. So far, the only possible way for him to combine these two preferences is to concatenate them in a certain order and view the result as a new atomic preference. However, neither $\Psi_1 \lhd \Psi_2$ nor $\Psi_2 \lhd \Psi_1$ meets the desired criteria – as they simply state that Ψ_1 is more relevant than Ψ_2 (or vice versa). The only way to accomplish the desired effect is to decompose Ψ_1 and Ψ_2 and rebuild a more complex atomic preference. This shows that the agent might have to define a new atomic preference for his newly introduced preference. □

The above discussion shows that it is necessary to provide additional ways for combining atomic preferences. This is the topic of this sub-section. We will introduce

[6] For brevity, we omit the description of the states s_i's.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

general preferences, which can be used to describe a multi-dimensional order among preferences. Formally, we define general preferences as follows.

Definition 10 (*General Preferences*)
A *general preference formula* is a formula satisfying one of the following conditions:

- An atomic preference Ψ is a general preference;
- If Ψ_1, Ψ_2 are general preferences, then $\Psi_1 \& \Psi_2$, $\Psi_1 \mid \Psi_2$, and $!\,\Psi_1$ are general preferences;
- If $\Psi_1, \Psi_2, \ldots, \Psi_k$ is a collection of general preferences, then $\Psi_1 \lhd \Psi_2 \lhd \cdots \lhd \Psi_k$ is a general preference.

In the above definition, the operators $\&, |, !$ are used to express different ways to combine preferences. Syntactically, they are similar to the operations $\wedge, \vee, \neg$ employed in the construction of basic desire formulae. Semantically, they differ from the operations $\wedge, \vee, \neg$ in a subtle way. Intuitively, the constituents of a general preference are atomic preferences; and a general preference provides a means for combining different orderings among trajectories created by its constituents, thus providing a means for the selection of a most preferred trajectory. Let us consider the case where a general preference has two constituents Ψ_1 and Ψ_2. As we will see later, each preference will induce two relations on trajectories, the indistinguishable and preferred relations, as in the case of atomic preferences. In other words, Ψ_i can be represented by this pair of relations. Given a general preference Ψ_i, let o_i and e_i denote the set of pairs of trajectories (α, β) such that $\alpha \prec_{\Psi_i} \beta$ and $\alpha \approx_{\Psi_i} \beta$, respectively. The operators $\&, |, !$ look at this characterization and define two relations among trajectories that satisfy the following equations:

- For the formula $\Psi_1 \& \Psi_2$, we define

$$(o_1, e_1) \& (o_2, e_2) \stackrel{\text{def}}{=} ((o_1 \cap o_2), (e_1 \cap e_2)).$$

  This says that the relation representing the ordering between trajectories induced by & is created as the intersection of the relations representing the orderings between trajectories induced by its components. In this case, a most preferred trajectory is the one which is most preferred w.r.t. every component of the formula.
- For the formula $\Psi_1 \mid \Psi_2$, we define

$$(o_1, e_1) \mid (o_2, e_2) \stackrel{\text{def}}{=} ((o_1 \cap o_2) \cup (o_1 \cap e_2) \cup (e_1 \cap o_2), (e_1 \cap e_2)).$$

  Here, the relation induced by | guarantees that the most preferred trajectory is the one which is most preferred w.r.t. at least one component of the formula and it is indistinguishable from others w.r.t. the remaining component of the formula.
- For the formula $!\Psi_1$, we define

$$!(o_1, e_1) \stackrel{\text{def}}{=} ((o_1{}^{-1} \cup e_1), e_1)$$

  which basically reverses the relations induced by Ψ_1.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

This is made more precise in the following definition.

Definition 11 (*Ordering Between Trajectories w.r.t. General Preferences*)
Let Ψ be a general preference and let α, β be two trajectories. We say that

- α is *preferred* to β, denoted by $\alpha \prec_\Psi \beta$, if:
  1. Ψ is an atomic preference and $\alpha \prec_\Psi \beta$
  2. $\Psi = \Psi_1 \& \Psi_2$ and $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$
  3. $\Psi = \Psi_1 \mid \Psi_2$ and:
     (a) $\alpha \prec_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2} \beta$; or
     (b) $\alpha \prec_{\Psi_2} \beta$ and $\alpha \approx_{\Psi_1} \beta$; or
     (c) $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$
  4. $\Psi = !\Psi_1$ and $\beta \prec_{\Psi_1} \alpha$
  5. $\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$, and there exists $1 \leqslant i \leqslant k$ such that: *(i)* $\forall (1 \leqslant j < i)$ we have that $\alpha \approx_{\Psi_j} \beta$, and *(ii)* $\alpha \prec_{\Psi_i} \beta$.
- α is *indistinguishable* from β, denoted by $\alpha \approx_\Psi \beta$, if:
  1. Ψ is an atomic preference and $\alpha \approx_\Psi \beta$.
  2. $\Psi = \Psi_1 \& \Psi_2$, $\alpha \approx_{\Psi_1} \beta$, $\alpha \approx_{\Psi_2} \beta$.
  3. $\Psi = \Psi_1 \mid \Psi_2$, $\alpha \approx_{\Psi_1} \beta$, and $\alpha \approx_{\Psi_2} \beta$.
  4. $\Psi = !\Psi_1$ and $\alpha \approx_{\Psi_1} \beta$.
  5. $\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$, and for all $1 \leqslant i \leqslant k$ we have that $\alpha \approx_{\Psi_i} \beta$.

Similar as above, $\alpha \preceq_\Psi \beta$ indicates that either $\alpha \prec_\Psi \beta$ or $\alpha \approx_\Psi \beta$. Before we move on, let us observe that the formula $\Psi_1 \lhd \ldots \lhd \Psi_n$, where each Ψ_i is a basic desire, can be viewed as both an atomic preference as well as a general preference. This is not ambiguous since the semantic definition for the two cases coincide. It is easy to see that the following holds for $\preceq_\Psi$.

Lemma 2
For every pair of trajectories α and β and a general preference Ψ such that $\alpha \preceq_\Psi \beta$,

- if $\Psi = \Psi_1 \& \Psi_2$ then $\alpha \preceq_{\Psi_1} \beta$ and $\alpha \preceq_{\Psi_2} \beta$.
- if $\Psi = \Psi_1 \mid \Psi_2$ then $\alpha \preceq_{\Psi_1} \beta$ and $\alpha \preceq_{\Psi_2} \beta$.
- if $\Psi = !\Psi_1$ then $\beta \preceq_{\Psi_1} \alpha$.

We can also show that for every general preference Ψ, $\approx_\Psi$ is an equivalence relation over trajectories and $\preceq_\Psi$ is a partial order on the set of representatives of the equivalence classes of $\approx_\Psi$. To prove this property, we need the following lemma.

Lemma 3
Let Ψ be a general preference formula and let α, β, and γ be three trajectories. It holds that

- if $\alpha \prec_\Psi \beta$ and $\beta \approx_\Psi \gamma$ then $\alpha \prec_\Psi \gamma$; and
- if $\alpha \prec_\Psi \beta$ and $\alpha \approx_\Psi \gamma$ then $\gamma \prec_\Psi \beta$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Proof
Let us prove the result by structural induction on Ψ. Because the proof of the two items are almost the same, we present below the proof of the first item.

- *Base:* If Ψ is an atomic preference, and $\Psi = \varphi_1 \lhd \cdots \lhd \varphi_k$ then from $\alpha \prec_\Psi \beta$ (Definition 8) we obtain that there exists $1 \leqslant i \leqslant k$ such that
  — $\alpha \approx_{\varphi_j} \beta$ for all $1 \leqslant j < i$ and
  — $\alpha \prec_{\varphi_i} \beta$.

  Furthermore, from $\beta \approx_\Psi \gamma$ we have that $\beta \approx_{\varphi_j} \gamma$ for all $1 \leqslant j \leqslant k$. Since $\approx_{\varphi_j}$ are all equivalence relations, we obtain that $\alpha \approx_{\varphi_j} \gamma$ for all $1 \leqslant j < i$; furthermore, since $\alpha \prec_{\varphi_i} \beta$, then $\alpha \models \varphi_i$ and $\beta \not\models \varphi_i$. Since $\beta \approx_{\varphi_i} \gamma$ then necessarily we have also that $\gamma \not\models \varphi_i$. This allows us to conclude that $\alpha \prec_\Psi \gamma$.
- *Inductive Step:*
  1. $\Psi = \Psi_1 \& \Psi_2$. Since $\alpha \prec_\Psi \beta$, we have $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$. Furthermore, $\beta \approx_\Psi \gamma$ implies $\beta \approx_{\Psi_1} \gamma$ and $\beta \approx_{\Psi_2} \gamma$. From the induction hypothesis we have $\alpha \prec_{\Psi_1} \gamma$ and $\alpha \prec_{\Psi_2} \gamma$, which leads to $\alpha \prec_\Psi \gamma$.
  2. $\Psi = \Psi_1 \mid \Psi_2$. From $\alpha \prec_\Psi \beta$ we have three possible cases:
     (a) $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$. In this case, since $\beta \approx_\Psi \gamma$ implies $\beta \approx_{\Psi_1} \gamma$ and $\beta \approx_{\Psi_2} \gamma$, we have that $\alpha \prec_{\Psi_1} \gamma$ and $\alpha \prec_{\Psi_2} \gamma$. This implies that $\alpha \prec_\Psi \gamma$.
     (b) $\alpha \prec_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2} \beta$. From $\beta \approx_\Psi \gamma$ we obtain $\beta \approx_{\Psi_1} \gamma$ and $\beta \approx_{\Psi_2} \gamma$. Since $\approx_{\Psi_2}$ is an equivalence relation, we can infer $\alpha \approx_{\Psi_2} \gamma$. Furthermore, from the induction hypothesis we obtain $\alpha \prec_{\Psi_1} \gamma$. These two conclusions lead to $\alpha \prec_\Psi \gamma$.
     (c) $\alpha \prec_{\Psi_2} \beta$ and $\alpha \approx_{\Psi_1} \beta$. This case is symmetrical to the previous one.
  3. $\Psi = !\Psi_1$. From $\alpha \prec_\Psi \beta$ we obtain $\beta \prec_{\Psi_1} \alpha$. Since $\beta \approx_\Psi \gamma$ implies $\beta \approx_{\Psi_1} \gamma$, from the induction hypothesis we can conclude $\gamma \prec_{\Psi_1} \alpha$ and ultimately $\alpha \prec_\Psi \gamma$.
  4. $\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$. From the definition of $\alpha \prec_\Psi \beta$ we have that there exists $1 \leqslant i \leqslant k$ such that
     (a) $\alpha \approx_{\Psi_j} \beta$ for all $1 \leqslant j < i$ and
     (b) $\alpha \prec_{\Psi_i} \beta$

     Furthermore, from $\beta \approx_\Psi \gamma$ we know that $\beta \approx_{\Psi_j} \gamma$ for all $1 \leqslant j \leqslant k$. Since $\approx_{\Psi_j}$ are all equivalence relations, we have that $\alpha \approx_{\Psi_j} \gamma$ for all $1 \leqslant j < i$. Furthermore, from the induction hypothesis, from $\alpha \prec_{\Psi_i} \beta$ and $\beta \approx_{\Psi_i} \gamma$ we can conclude $\alpha \prec_{\Psi_i} \gamma$. This finally leads to $\alpha \prec_\Psi \gamma$.

□

Lemma 4
Let Ψ be a general preference formula and α, β, and γ be three trajectories. It holds that if $\alpha \prec_\Psi \beta$ and $\beta \prec_\Psi \gamma$ then $\alpha \prec_\Psi \gamma$.

Proof
Let us prove the result by structural induction on Ψ.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- *Base:* If Ψ is an atomic preference, and $\Psi = \varphi_1 \lhd \cdots \lhd \varphi_k$ then from $\alpha \prec_\Psi \beta$ (Definition 8) we obtain that there exists $1 \leqslant i \leqslant k$ such that
  - $\alpha \approx_{\varphi_j} \beta$ for all $1 \leqslant j < i$ and
  - $\alpha \prec_{\varphi_i} \beta$.

  Furthermore, from $\beta \prec_\Psi \gamma$, we know that there exists $1 \leqslant l \leqslant k$ such that
  - $\beta \approx_{\varphi_j} \gamma$ for all $1 \leqslant j < l$ and
  - $\beta \prec_{\varphi_l} \gamma$.

  If $i \leqslant l$, it is easy to see that $\alpha \approx_{\varphi_j} \gamma$ for $j < i$ and $\alpha \prec_{\varphi_i} \gamma$, which implies that $\alpha \prec_\Psi \gamma$. If $l < i$ holds, we have that $\alpha \approx_{\varphi_j} \gamma$ for all $1 \leqslant j < l$ and $\alpha \prec_{\varphi_i} \gamma$. Thus, $\alpha \prec_\Psi \gamma$.
- *Inductive Step:*
  - $\Psi = \Psi_1 \& \Psi_2$. Since $\alpha \prec_\Psi \beta$ then we have $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$. Furthermore, $\beta \prec_\Psi \gamma$ implies $\beta \prec_{\Psi_1} \gamma$ and $\beta \prec_{\Psi_2} \gamma$. From the induction hypothesis we have $\alpha \prec_{\Psi_1} \gamma$ and $\alpha \prec_{\Psi_2} \gamma$, which leads to $\alpha \prec_\Psi \gamma$.
  - $\Psi = \Psi_1 \mid \Psi_2$. From $\alpha \prec_\Psi \beta$ we have three possible cases:
    1. $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$. The proof for this case is similar to the case In this case, since $\beta \prec_\Psi \gamma$ implies $\beta \prec_{\Psi_1} \gamma$ and $\beta \prec_{\Psi_2} \gamma$, then we have $\alpha \prec_{\Psi_1} \gamma$ and $\alpha \prec_{\Psi_2} \gamma$. This implies that $\alpha \prec_\Psi \gamma$.
    2. $\alpha \prec_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2} \beta$. From $\beta \approx_\Psi \gamma$ we obtain $\beta \approx_{\Psi_1} \gamma$ and $\beta \approx_{\Psi_2} \gamma$. Since $\approx_{\Psi_2}$ is an equivalence relation, we can infer $\alpha \approx_{\Psi_2} \gamma$. Furthermore, from the induction hypothesis we obtain $\alpha \prec_{\Psi_1} \gamma$. These two conclusions lead to $\alpha \prec_\Psi \gamma$.
    3. $\alpha \prec_{\Psi_2} \beta$ and $\alpha \approx_{\Psi_1} \beta$. This case is symmetrical to the previous one.
  - $\Psi = !\Psi_1$. From $\alpha \prec_\Psi \beta$ we obtain $\beta \prec_{\Psi_1} \alpha$. Since $\beta \approx_\Psi \gamma$ implies $\beta \approx_{\Psi_1} \gamma$, from the induction hypothesis we can conclude $\gamma \prec_{\Psi_1} \alpha$ and ultimately $\alpha \prec_\Psi \gamma$.
  - $\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$. From the definition of $\alpha \prec_\Psi \beta$ we have that there exists $1 \leqslant i \leqslant k$ such that
    - $\alpha \approx_{\Psi_j} \beta$ for all $1 \leqslant j < i$ and
    - $\alpha \prec_{\Psi_i} \beta$

    Furthermore, from $\beta \approx_\Psi \gamma$ we know that $\beta \approx_{\Psi_j} \gamma$ for all $1 \leqslant j \leqslant k$. Since all $\approx_{\Psi_j}$ are equivalence relations, then we have that $\alpha \approx_{\Psi_j} \gamma$ for all $1 \leqslant j < i$. Furthermore, from the induction hypothesis, from $\alpha \prec_{\Psi_i} \beta$ and $\beta \approx_{\Psi_i} \gamma$ we can conclude $\alpha \prec_{\Psi_i} \gamma$. This finally leads to $\alpha \prec_\Psi \gamma$.

□

We now show that $\preceq_\Psi$ is a partial order on the set of representatives of the equivalence classes of $\approx_\Psi$.

Proposition 4

Let Ψ be a general preference. Then $\preceq_\Psi$ is a partial order on the set of representatives of the equivalence classes of $\approx_\Psi$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Proof

We need to show that $\preceq_\Psi$ is reflective, antisymmetric, and transitive. Reflexivity follows from the fact that $\approx_\Phi$ is an equivalence relation and thus $\alpha \preceq_\Psi \alpha$ holds for every α. We prove that $\preceq_\Psi$ is antisymmetric and transitive using structural induction on Ψ.

- *Base:* This corresponds to Ψ being an atomic preference. The two properties follow from Proposition 3.
- *Inductive step:* Let us consider the possible cases for Ψ.
  1. $\Psi = \Psi_1 \& \Psi_2$.
     (a) *Anti-symmetry:* consider two representatives α, β and let us assume that $\alpha \preceq_\Psi \beta$ and $\beta \preceq_\Psi \alpha$. Again, it is enough if we can show that $\alpha \approx_\Psi \beta$. $\alpha \preceq_\Psi \beta$ implies that $\alpha \preceq_{\Psi_1} \beta$ and $\alpha \preceq_{\Psi_2} \beta$ (Lemma 2). $\beta \preceq_\Psi \alpha$ implies that $\beta \preceq_{\Psi_1} \alpha$ and $\beta \preceq_{\Psi_2} \alpha$ (Lemma 2). By the induction hypothesis, we have that $\alpha \approx_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2} \beta$ which imply that $\alpha \approx_\Psi \beta$.
     (b) *Transitivity:* consider three representatives α_1, α_2, and α_3 with $\alpha_1 \preceq_\Psi \alpha_2$ and $\alpha_2 \preceq_\Psi \alpha_3$. The first relationship implies that $\alpha_1 \preceq_{\Psi_1} \alpha_2$ and $\alpha_1 \preceq_{\Psi_2} \alpha_2$. The second relationship implies that $\alpha_2 \preceq_{\Psi_1} \alpha_3$, and $\alpha_2 \preceq_{\Psi_2} \alpha_3$. From the induction hypothesis we have $\alpha_1 \preceq_{\Psi_1} \alpha_3$ and $\alpha_1 \preceq_{\Psi_2} \alpha_3$. Thus, $\alpha_1 \preceq_\Psi \alpha_3$.
  2. $\Psi = \Psi_1 \mid \Psi_2$. Similar arguments to the above case (with the help of Lemma 2) allows us to conclude that anti-symmetry and transitivity also holds for this case.
  3. $\Psi = !\Psi_1$.
     (a) *Anti-symmetry:* consider two representatives α, β and let us assume that $\alpha \preceq_\Psi \beta$ and $\beta \preceq_\Psi \alpha$. Lemma 2 imply that $\beta \preceq_{\Psi_1} \alpha$ and $\alpha \preceq_{\Psi_1} \beta$. By the induction hypothesis, we have that $\alpha \approx_{\Psi_1} \beta$. This allows us to conclude that $\alpha \approx_\Psi \beta$.
     (b) *Transitivity:* consider three representatives α_1, α_2, and α_3 with $\alpha_1 \preceq_\Psi \alpha_2$ and $\alpha_2 \preceq_\Psi \alpha_3$. Again, Lemma 2 implies that $\alpha_2 \preceq_{\Psi_1} \alpha_1$ and $\alpha_3 \preceq_{\Psi_1} \alpha_2$. The induction hypothesis implies that $\alpha_3 \preceq_{\Psi_1} \alpha_1$, and hence, $\alpha_1 \preceq_\Psi \alpha_3$.
  4. $\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$.
     (a) *Anti-symmetry:* consider two representatives α, β with $\alpha \preceq_\Psi \beta$ and $\beta \preceq_\Psi \alpha$. Assume that $\alpha \prec_\Psi \beta$. This means that there exists $1 \leqslant i \leqslant k$ such that $\alpha \approx_{\Psi_j} \beta$ for all $1 \leqslant j < i$ and $\alpha \prec_{\Psi_i} \beta$. This will imply that $\beta \preceq_\Psi \alpha$ cannot hold, i.e., we have a contradiction. This means that $\alpha \approx_\Psi \beta$.
     (b) *Transitivity:* consider three representatives α_1, α_2, and α_3 with $\alpha_1 \preceq_\Psi \alpha_2$ and $\alpha_2 \preceq_\Psi \alpha_3$. We have four sub-cases:
        i $\alpha_1 \approx_\Psi \alpha_2$ and $\alpha_2 \approx_\Psi \alpha_3$. In this case, we have that $\alpha_1 \approx_\Psi \alpha_3$ because $\approx_\Psi$ is an equivalence relation.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

ii $\alpha_1 \prec_\Psi \alpha_2$ and $\alpha_2 \approx_\Psi \alpha_3$. Lemma 3 implies that $\alpha_1 \prec_\Psi \alpha_3$.

iii $\alpha_1 \approx_\Psi \alpha_2$ and $\alpha_2 \prec_\Psi \alpha_3$. Lemma 3 implies that $\alpha_1 \prec_\Psi \alpha_3$.

iv $\alpha_1 \prec_\Psi \alpha_2$ and $\alpha_2 \prec_\Psi \alpha_3$. This implies that there exists $1 \leqslant i_1, i_2 \leqslant k$ such that $\alpha_1 \approx_{\Psi_j} \alpha_2$ for all $1 \leqslant j < i_1$ and $\alpha_1 \prec_{\Psi_{i_1}} \alpha_2$; and $\alpha_2 \approx_{\Psi_j} \alpha_3$ for all $1 \leqslant j < i_2$ and $\alpha_2 \prec_{\Psi_{i_2}} \alpha_3$.
If $i_1 < i_2$ then from the fact that $\approx_{\Psi_j}$ are equivalence relations, we can conclude that $\alpha_1 \approx_{\Psi_j} \alpha_3$ for all $1 \leqslant j < i_1$. Furthermore, by Lemma 3, from $\alpha_1 \prec_{\Psi_{i_1}} \alpha_2$ and $\alpha_2 \approx_{\Psi_{i_1}} \alpha_3$, we can conclude that $\alpha_1 \prec_{\Psi_{i_1}} \alpha_3$. This leads to $\alpha_1 \prec_\Psi \alpha_3$.
Similarly, if $i_1 > i_2$, we have that $\alpha_1 \approx_{\Psi_j} \alpha_3$ for all $1 \leqslant j < i_2$ and $\alpha_1 \prec_{\Psi_{i_2}} \alpha_3$. This leads to $\alpha_1 \prec_\Psi \alpha_3$.
Finally, if $i_1 = i_2$, then we have that $\alpha_1 \approx_{\Psi_j} \alpha_3$ for all $1 \leqslant j < i_1$. Furthermore, since $\alpha_1 \prec_{\Psi_i} \alpha_2$ and $\alpha_2 \prec_{\Psi_i} \alpha_3$, from the induction hypothesis we obtain $\alpha_1 \prec_{\Psi_i} \alpha_3$. This also leads to $\alpha_1 \prec_\Psi \alpha_3$.

□

Definition 12
Given a general preference Ψ, we say that a trajectory α is *most preferred* if there is no trajectory that is preferred to α.

The next example highlights some differences and similarities between basic desires and general preferences.

Example 7
Let us consider the original action theory presented in Section 2 with the action *buy_coffee* and a user having the goal of being at the school and having coffee. Intuitively, every trajectory achieving the goal of the user would require the action of going to the coffee shop, buying the coffee, and going to the school thereafter.

Let us consider the following two preferences (similar to those discussed in Example 4):

$$time = \textbf{always}(occ(buy_coffee) \vee (take_taxi <^e (drive \vee bus) <^e walk))$$

and

$$cost = \textbf{always}(occ(buy_coffee) \vee (walk <^e (drive \vee bus) <^e take_taxi)).$$

It is easy to see that most preferred trajectories with respect to *time* should contain only the actions *buy_coffee* and *take_taxi* while most preferred trajectories with respect to *cost* should contain only the actions *buy_coffee* and *walk*.

Consider the two preferences:

$$\Psi_1 = time \wedge cost$$

and

$$\Psi_2 = time \;\&\; cost.$$

Observe that Ψ_1 is a basic desire while Ψ_2 is a general preference. It is easy to see that there are no trajectories satisfying the preference Ψ_1. Thus, every trajectory

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

achieving the goal is a most preferred trajectory w.r.t. Ψ_1. At the same time, we can show that for every pair of trajectories α and β, neither $\alpha \prec_{\Psi_2} \beta$ nor $\beta \prec_{\Psi_2} \alpha$ holds.

Let us now consider the two preferences:

$$\Psi_3 = time \vee cost$$

and

$$\Psi_4 = time \mid cost$$

with respect to the same set of trajectories. Here, Ψ_3 is a basic desire while Ψ_4 is a general preference. We can see that any trajectory containing the actions *taxi* and *walk* would be most preferred with respect to Ψ_3. All of these trajectories are indistinguishable. For example, the trajectory

$$\alpha = s_0\ walk(home, coffee_shop)\ s_1\ buy_coffee\ s_2\ walk(coffee_shop, school)\ s_3$$

and the trajectory

$$\beta = s_0\ walk(home, coffee_shop)\ s'_1\ buy_coffee\ s'_2\ take_taxi(coffee_shop, school)\ s'_3$$

are indistinguishable with respect to Ψ_3. On the other hand, we have that $\alpha \prec_{cost} \beta$ (the minimal cost action is always used) and $\alpha \approx_{time} \beta$ (the fastest action is not used every time). This implies that $\alpha \prec_{\Psi_4} \beta$.

Let us consider now the preference

$$\Psi_5 = !\ time.$$

It is easy to see that Ψ_5 is equivalent to *cost* in the sense that every most preferred trajectory w.r.t. Ψ_5 is a most preferred trajectory w.r.t *cost* and vice versa. □

The next proposition is of interest for the computation of preferred trajectories[7].

Proposition 5
Let Ψ_1, Ψ_2 and Ψ_3 be three general preferences, $\Psi = \Psi_1 \lhd \Psi_2 \lhd \Psi_3$, and $\Gamma = \Psi_1 \lhd (\Psi_2 \lhd \Psi_3)$. For arbitrary trajectories α and β, the following holds:

- $\alpha \approx_\Psi \beta$ if and only if $\alpha \approx_\Gamma \beta$; and
- $\alpha \prec_\Psi \beta$ if and only if $\alpha \prec_\Gamma \beta$.

Proof

- We have that $\alpha \approx_\Psi \beta$ iff $\alpha \approx_{\Psi_i} \beta$ for $i \in \{1, 2, 3\}$ iff $\alpha \approx_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2 \lhd \Psi_3} \beta$ iff $\alpha \approx_\Gamma \beta$.
- Since $\alpha \prec_\Psi \beta$ iff there exists $i \in \{1, 2, 3\}$ such that $\alpha \approx_{\Psi_j} \beta$ for $1 \leqslant j < i$ and $\alpha \prec_{\Psi_i} \beta$, we have three cases: $i = 1$, 2, or 3. We consider these three cases:
  - $i = 1$. This implies immediately that $\alpha \prec_\Gamma \beta$.
  - $i = 2$. This means that $\alpha \approx_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$. This in turn implies that $\alpha \approx_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2 \lhd \Psi_3} \beta$, i.e., $\alpha \prec_\Gamma \beta$.
  - $i = 3$. This is similar to the case $i = 2$.

  Thus, we have that if $\alpha \prec_\Psi \beta$ then $\alpha \prec_\Gamma \beta$. The proof of the reverse is similar. □

[7] We would like to thank the anonymous reviewer who pointed out that this proposition is necessary for the computation presented in the next section.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

4 Computing preferred trajectories

In this section, we address the problem of computing preferred trajectories. The ability to use the operators $\wedge, \neg, \vee$ as well as $\&, |, !$ in the construction of preference formulae allows us to combine several preferences into a preference formula. For example, if a user has two atomic preferences Ψ and Φ, but does not prefer Ψ over Φ or vice versa, he can combine them in to a single preference $\Psi \wedge \Phi \lhd \Psi \vee \Phi \lhd \neg\Psi \wedge \neg\Phi$. The same can be done if Ψ or Φ are general preferences. Thus, without loss of generality, we can assume that we only have one preference formula to deal with. We would like to note that this way of combination of preferences creates a preference whose size could be exponential in the number of preferences. We believe, however, that it is more likely that a user – when presented with a set of preferences – will have a preferred order on some of these preferences. This information can be used to create a single preference with a reasonable and manageable size.

Given a planning problem $\langle D, I, G\rangle$ and a preference formula φ, we are interested in finding a most preferred trajectory α achieving G w.r.t. the preference φ. We will show how this can be done in answer set programming.

A naive encoding could be realized by modeling $\langle D, I, G\rangle$ in logic programming (following the scheme described in Son *et al.* (2005)), using an answer set engine to determine its answer sets – and, thus, the trajectories – and then filtering them according to φ.

In the rest of this section, instead, we present a more sophisticated approach which allows us to determine *a* most preferred trajectory. We achieve this by encoding each basic desire φ as a set of rules Π_φ and by developing two sets of rules Π_{sat} and Π_{pref}. The program Π_{sat} checks whether a basic desire is satisfied by a trajectory. On the other hand, Π_{pref} consists of rules that, when used together with the **maximize** construct of **smodels**, allow us to find a most preferred trajectory with respect to a preference formula. Since $\Pi(D, I, G)$ has already been discussed in Section 2, we will start by defining Π_φ.

4.1 Π_φ: Encoding of basic desire formulae

The encoding of a basic desire formula is similar to the encoding of a fluent formula proposed in Son *et al.* (2005). In our encoding, we will use the predicate *desire* as a domain predicate. The elements of the set $\{desire(l) \mid l \text{ is a fluent literal}\}$ belong to Π_φ. Each atom in this set declares the fact that each literal is also a desire. Next, we associate to each basic desire formula φ a unique name n_φ. If φ is a basic desire formula then it will be encoded as a set of facts, denoted by Π_φ. This set is defined inductively over the structure of φ[8]. The encoding is performed as follows.

- If φ is a fluent literal l then $\Pi_\varphi = \{desire(l)\}$;
- If $\varphi = goal(\phi)$ then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), goal(n_\phi)\}$;

[8] To simplify this encoding, we have developed a Prolog program that translates φ into Π_φ. This program can be downloaded from `http://www.cs.nmsu.edu/~tson/ASPlan/Preferences/conv_form.pl`.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- If $\varphi = occ(a)$ then $\Pi_\varphi = \{desire(n_\varphi), happen(n_\varphi, a)\}$;
- If $\varphi = \varphi_1 \wedge \varphi_2$ then $\Pi_\varphi = \Pi_{\varphi_1} \cup \Pi_{\varphi_2} \cup \{desire(n_\varphi), and(n_\varphi, n_{\varphi_1}, n_{\varphi_2})\}$;
- If $\varphi = \varphi_1 \vee \varphi_2$ then $\Pi_\varphi = \Pi_{\varphi_1} \cup \Pi_{\varphi_2} \cup \{desire(n_\varphi), or(n_\varphi, n_{\varphi_1}, n_{\varphi_2})\}$;
- If $\varphi = \neg\phi$ then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), negation(n_\varphi, n_\phi)\}$;
- If $\varphi = \textbf{next}(\phi)$ then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), next(n_\varphi, n_\phi)\}$;
- If $\varphi = \textbf{until}(\varphi_1, \varphi_2)$ then $\Pi_\varphi = \Pi_{\varphi_1} \cup \Pi_{\varphi_2} \cup \{desire(n_\varphi), until(n_\varphi, n_{\varphi_1}, n_{\varphi_2})\}$;
- If $\varphi = \textbf{always}(\phi)$ then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), always(n_\varphi, n_\phi)\}$;
- If $\varphi = \textbf{eventually}(\phi)$ then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), eventually(n_\varphi, n_\phi)\}$.

It is worth noting that, due to the uniqueness of names for basic desires, n_φ will not occur in $\Pi_\Phi \setminus \{desire(n_\varphi)\}$.

4.2 Π_{sat}: *Rules for checking of basic desire formula satisfaction*

We now present the set of rules that checks whether a trajectory satisfies a basic desire formula. Recall that an answer set of the program $\Pi(D, I, G)$ will contain a trajectory where action occurrences are recorded by atoms of the form $occ(a, t)$ and the truth value of fluent literals is represented by atoms of the form $holds(f, t)$, where $a \in \mathbf{A}$, f is a fluent literal, and t is a time moment between 0 and *length*. The program Π_{sat} defines the predicate $satisfy(F, T)$, where F and T are variables representing a basic desire and a time moment, respectively. The satisfiability of a fluent formula at a time moment will be defined by the predicate $h(F, T)$ – which builds on the previous mentioned predicate *holds* and the usual logical operators, such as $\wedge, \vee, \neg$. Intuitively, $satisfy(F, T)$ says that the basic desire F is satisfied by the trajectory starting from the time moment T. The rules of Π_{sat} are defined inductively on the structure of F and are given next.

$$satisfy(F,T) \leftarrow desire(F), goal(F), satisfy(F, length). \quad (1)$$

$$satisfy(F,T) \leftarrow desire(F), happen(F, A), occ(A, T). \quad (2)$$

$$satisfy(F,T) \leftarrow desire(F), literal(F), holds(F, T). \quad (3)$$

$$satisfy(F,T) \leftarrow desire(F), and(F, F_1, F_2), satisfy(F_1, T), satisfy(F_2, T). \quad (4)$$

$$satisfy(F,T) \leftarrow desire(F), or(F, F_1, F_2), satisfy(F_1, T). \quad (5)$$

$$satisfy(F,T) \leftarrow desire(F), or(F, F_1, F_2), satisfy(F_2, T). \quad (6)$$

$$satisfy(F,T) \leftarrow desire(F), negation(F, F_1), not\ satisfy(F_1, T). \quad (7)$$

$$satisfy(F,T) \leftarrow desire(F), until(F, F_1, F_2), T < T_1, during(F_1, T, T_1 - 1), satisfy(F_2, T_1). \quad (8)$$

$$satisfy(F,T) \leftarrow desire(F), until(F, F_1, F_2), satisfy(F_2, T). \quad (9)$$

$$satisfy(F,T) \leftarrow desire(F), always(F, F_1), during(F_1, T, length). \quad (10)$$

$$satisfy(F,T) \leftarrow desire(F), next(F, F_1), satisfy(F_1, T + 1). \quad (11)$$

$$satisfy(F,T) \leftarrow desire(F), eventually(F, F_1), T \leqslant T1, satisfy(F_1, T1). \quad (12)$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

$$during(F,T,T_1) \leftarrow T < T_1, desire(F), satisfy(F,T), during(F,T+1,T_1). \quad (13)$$

$$during(F,T,T) \leftarrow desire(F), satisfy(F,T). \quad (14)$$

In the next theorem, we prove the correctness of Π_{sat}. We need some additional notation. For a trajectory $\alpha = s_0 a_1 \dots a_n s_n$, let

$$\alpha^{-1} = \{occ(a_i, i-1) \mid i \in \{1,\dots,n\}\} \cup \{holds(f,i) \mid f \in s_i, i \in \{0,\dots,n\}\}.$$

We have:

Theorem 2
Let $\langle D, I, G\rangle$ be a planning problem, $\alpha = s_0 a_1 \dots a_n s_n$ be a trajectory, and φ be a basic desire formula. Then, for every t, $0 \leqslant t \leqslant length$ and every basic desire formula η with $desire(n_\eta) \in \Pi_\varphi$,

$$\alpha[t] \models \eta \quad \text{iff} \quad \Pi_\varphi \cup \Pi_{sat} \cup (\alpha)^{-1} \models satisfy(n_\eta, t).$$

(Recall that $\alpha[t]$ is the trajectory $s_t a_{t+1} \dots a_n s_n$.)

Proof
First, we prove that the program $\Pi = \Pi_\varphi \cup \Pi_{sat} \cup (\alpha)^{-1}$ has only one answer set. It is well-known that if a program is locally stratified then it has a unique answer set (Apt *et al.* 1988; Przymusinski 1988). We will show that Π (more precisely, the set of ground instances of rules in Π) is indeed locally stratified. To accomplish this we need to find a mapping λ from literals of the grounding of Π to **N** that has the property: if

$$A_0 \leftarrow A_1, A_2, \dots A_n, not\ B_1, not\ B_2, \dots not\ B_m$$

is a rule in Π, then $\lambda(A_0) \geqslant \lambda(A_i)$ for all $1 \leqslant i \leqslant n$ and $\lambda(A_0) > \lambda(B_j)$ for all $1 \leqslant j \leqslant m$.

To define λ, we first associate a non-negative number $\sigma(\phi)$ to each constant n_η as follows. Intuitively, $\sigma(\phi)$ represents the complexity of ϕ.

- $\sigma(l) = 0$ if l is a literal.
- $\sigma(n_\eta) = 0$ if η has the form $occ(a)$.
- $\sigma(n_\eta) = \sigma(n_{\eta_1}) + 1$ if η has the form $\neg\eta_1$, **next**(η_1), **eventually**(η_1), or **always**(η_1).
- $\sigma(n_\eta) = \max\{\sigma(n_{\eta_1}), \sigma(n_{\eta_2})\} + 1$ if η has the form $\eta_1 \wedge \eta_2$, $\eta_1 \vee \eta_2$, or **until**(η_1, η_2).
- $\sigma(n_\eta) = \sigma(n_{\eta_1})$ if $\eta =$ **goal**(η_1).

We define λ as follows.

- $\lambda(satisfy(n_\eta, t)) = 5 * \sigma(\eta) + 2$,
- $\lambda(during(n_\eta, t, t')) = 5 * \sigma(\eta) + 4$, and
- $\lambda(l) = 0$ for every other literal of Π.

We need to check that λ satisfies the necessary requirements. For example, for the rule (1), we have that $\lambda(satisfy(n_\eta, t)) = \lambda(satisfy(n_\eta, length)) = 5 * \sigma(n_\eta) + 2$ and $\lambda(satisfy(n_\eta, t)) \geqslant 2 > 0 = \lambda(goal(n_\eta)) = \lambda(desire(n_\eta))$. For the rule (7), we have that $\lambda(satisfy(n_\eta, t)) = 5 * \sigma(n_\eta) + 2 = 5 * (\sigma(n_{\eta_1}) + 1) + 2 > 5 * (\sigma(n_{\eta_1})) + 2 = \lambda(satisfy(n_{\eta_1}, length))$. The verification of this property for other rules is similar.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Thus, we can conclude that Π has only one answer set. Let X be the answer set of Π. We prove the conclusion of the theorem by induction over $\sigma(n_\eta)$.

Base: Let η be a formula with $\sigma(n_\eta) = 0$. By the definition of σ, we know that η is a fluent literal or of the form $occ(a)$. If η is a literal, then η is true in s_t iff η is in s_t, that is, iff $holds(\eta, t)$ belongs to X, which, because of rule (3), proves the base case. If $\eta = occ(a)$ then we know that $happen(n_\eta, a) \in \Pi_\varphi$. Thus, $satisfy(n_\eta, t) \in X$ iff $occ(a, t) \in X$ (because of the rule (2)) iff $\alpha[t] \models \eta$.

Step: Assume that for all $0 \leqslant j \leqslant k$ and formula η such that $\sigma(n_\eta) = j$, $\alpha[t] \models \eta$ iff $satisfy(n_\eta, t)$ is in X.

Let η be such a formula that $\sigma(n_\eta) = k + 1$. Because of the definition of σ, we have the following cases:

- **Case 1:** $\eta = \neg\eta_1$. We have that $\sigma(n_{\eta_1}) = \sigma(n_\eta) - 1 = k$. Since $negation(n_\eta, n_{\eta_1}) \in X$, $satisfy(n_\eta, t) \in X$ iff the body of rule (7) is satisfied by X iff $satisfy(n_{\eta_1}, t) \notin X$ iff $\alpha[t] \not\models \eta_1$ (by the induction hypothesis) iff $\alpha[t] \models \eta$.
- **Case 2:** $\eta = \eta_1 \wedge \eta_2$. Similar to the first case, it follows from the rule (4) and the facts $desire(n_\eta)$ and $and(n_\eta, n_{\eta_1}, n_{\eta_2})$ in Π_φ that $satisfy(n_\eta, t) \in X$ iff the body of rule (4) is satisfied by X iff $satisfy(n_{\eta_1}, t) \in X$ and $satisfy(n_{\eta_2}, t) \in X$ iff $\alpha[t] \models \eta_1$ and $\alpha[t] \models \eta_2$ (induction hypothesis) iff $\alpha[t] \models \eta$.
- **Case 3:** $\eta = \eta_1 \vee \eta_2$. The proof is similar to the above cases, relying on the two rules (5), (6), and the facts that $desire(n_\eta) \in \Pi_\varphi$ and $or(n_\eta, n_{\eta_1}, n_{\eta_2}) \in \Pi_\varphi$.
- **Case 4:** $\eta = \textbf{until}(\eta_1, \eta_2)$. We have that $\sigma(n_{\eta_1}) \leqslant k$ and $\sigma(n_{\eta_2}) \leqslant k$. Assume that $\alpha[t] \models \eta$. By Definition 3, there exists $t \leqslant t_2 \leqslant n$ such that $\alpha[t_2] \models \eta_2$ and for all $t \leqslant t_1 < t_2$, $\alpha[t_1] \models \eta_1$. By the induction hypothesis, $satisfy(n_{\eta_2}, t_2) \in X$ and $satisfy(n_{\eta_1}, t_1) \in X$ for $t \leqslant t_1 < t_2$. It follows that $during(n_{\eta_1}, t, t_2 - 1) \in X$. Because of rule (8)-(9), we have $satisfy(n_\eta, t) \in X$. On the other hand, if $satisfy(n_\eta, t) \in X$, because the only rules supporting $satisfy(n_\eta, t)$ are (8)-(9), there exists t_2, $t \leqslant t_2 \leqslant length$, and $during(n_{\eta_1}, t, t_2 - 1) \in X$, and $satisfy(n_{\eta_2}, t_2)$. It follows from $during(n_{\eta_1}, t, t_2 - 1) \in X$ that $satisfy(n_{\eta_1}, t_1) \in X$ for all $t \leqslant t_1 < t_2$. By the induction hypothesis, we have $\alpha[t_1] \models \eta_1$ for all $t \leqslant t_1 < t_2$ and $\alpha[t_2] \models \eta_2$. Thus $\alpha[t] \models \textbf{until}(\eta_1, \eta_2)$, i.e., $\alpha[t] \models \eta$.
- **Case 5:** $\eta = \textbf{next}(\eta_1)$. Note that $\sigma(n_{\eta_1}) \leqslant k$. Rule (11) is the only rule supporting $satisfy(n_\eta, t)$ where $\eta = \textbf{next}(\eta_1)$. So $satisfy(n_\eta, t) \in X$ iff $satisfy(n_{\eta_1}, t + 1) \in X$ iff $\alpha[t + 1] \models \eta_1$ iff $\alpha[t] \models \textbf{next}(\eta_1)$.
- **Case 6:** $\eta = \textbf{always}(\eta_1)$. We note that $\sigma(n_{\eta_1}) \leqslant k$. Observe that $satisfy(n_\eta, t)$ is supported only by rule (10). So $satisfy(n_\eta, t) \in X$ iff $during(n_{\eta_1}, t, n) \in X$. The latter happens iff $satisfy(n_{\eta_1}, t_1) \in X$ for all $t \leqslant t_1 \leqslant n$, that is, iff $\alpha[t_1] \models \eta_1$ for all $t \leqslant t_1 \leqslant n$ which is equivalent to $\alpha[t] \models \textbf{always}(\eta_1)$, i.e., iff $\alpha[t] \models \eta$.
- **Case 7:** $\eta = \textbf{eventually}(\eta_1)$. We know that $satisfy(n_\eta, t) \in X$ is supported only by rule (12). So $satisfy(n_\eta, t) \in X$ iff there exists $t \leqslant t_1 \leqslant n$ such that $satisfy(n_{\eta_1}, t_1) \in X$. Because $\sigma(n_{\eta_1}) \leqslant k$, by induction, $satisfy(n_\eta, t) \in X$ iff there exists $t \leqslant t_1 \leqslant n$ such that $\alpha[t_1] \models \eta_1$, that is, iff $\alpha[t] \models \textbf{eventually}(\eta_1)$, i.e., iff $\alpha[t] \models \eta$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- **Case 8:** $\eta = \mathbf{goal}(\eta_1)$. Since η_1 does not contain the **goal** operator, it follows from the above cases that $satisfy(n_{\eta_1}, n) \in X$ iff $\alpha[n] \models \eta_1$. From the rule (1), we can conclude that $satisfy(n_\eta, t) \in X$ iff $satisfy(n_{\eta_1}, n) \in X$ iff $\alpha[t] \models \eta$.

The above cases prove the inductive step and, hence, the theorem. □

The next theorem follows from Theorems (1) and (2).

Theorem 3
Let $\langle D, I, G\rangle$ be a planning problem and φ be a basic desire formula. For every answer set M of the program $\Pi(D, I, G, \varphi) = \Pi(D, I, G) \cup \Pi_\varphi \cup \Pi_{sat}$,

$$\alpha_M \models \varphi \qquad \text{iff} \qquad satisfy(n_\varphi, 0) \in M.$$

where α_M denotes the trajectory achieving G represented by M.

Proof
Let S be the set of literals of the program $\Pi(D, I, G)$. It is easy to see that for every rule r in $\Pi(D, I, G, \varphi)$ if the head of r belongs to S then every literal occurring in the body of r also belongs to S. Thus, S is a splitting set of $\Pi(D, I, G, \varphi)$. Using the Splitting Theorem (Lifschitz and Turner 1994), we can show that M is an answer set of $\Pi(D, I, G, \varphi)$ iff $M = X \cup Y$, where X is an answer set of $\Pi(D, I, G)$ and Y is an answer set of the program $\Pi_\varphi \cup \Pi_{sat} \cup (\alpha_M)^{-1}$. It follows from Theorem 2 that $\alpha_M[0] \models \varphi$ iff $satisfy(n_\varphi, 0) \in Y$ iff $satisfy(n_\varphi, 0) \in M$. □

The above theorem allows us to compute a most preferred trajectory using the **smodels** system. Let $\Pi(D, I, G, \varphi)$ be the program consisting of the $\Pi(D, I, G) \cup \Pi_\varphi \cup \Pi_{sat}$ and the rule

$$\mathbf{maximize}\{satisfy(n_\varphi, 0) = 1, not\ satisfy(n_\varphi, 0) = 0\}. \tag{15}$$

Note that rule (15) represents the fact that the answer sets in which $satisfy(n_\varphi, 0)$ holds are most preferred. **smodels** will first try to compute answer sets of Π in which $satisfy(n_\varphi, 0)$ holds; only if no answer sets with this property exist, other answer sets will be considered.[9]

Theorem 4
Let $\langle D, I, G\rangle$ be a planning problem and φ be a basic desire formula. For every answer set M of $\Pi(D, I, G, \varphi)$, if $satisfy(n_\varphi, 0) \in M$ then α_M is a most preferred trajectory w.r.t. φ.

Proof
The result follows directly from Theorem 3: $satisfy(n_\varphi, 0) \in M$ implies that $\alpha_M \models \varphi$, hence α_M is a most preferred trajectory w.r.t. φ. □

The above theorem gives us a way to compute a most preferred trajectory with respect to a basic desire. We will now generalize this approach to deal with atomic and general preferences. The intuition is to associate to the different components of

[9] The current implementation of **smodels** has some restrictions on using the **maximize** construct; our **jsmodels** system can now deal with this construct properly.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

the preference formula a *weight*; these weights are then used to obtain a weight for each trajectory, based on what components of the preference formula are satisfied by the trajectory. The **maximize** construct in **smodels** can then be used to compute answer sets with maximal weight, thus computing most preferred trajectories. The weight functions are defined as follows.

Definition 13
Given a general preference Ψ, *a weight function* w_Ψ *w.r.t.* Ψ (or weight function, for short, when it is clear from the context what is Ψ) assigns to each trajectory α a non-negative number $w_\Psi(\alpha)$.

Since our goal is to use weight functions in generating most preferred trajectories using the **maximize** construct in **smodels**, we would like to find weight functions which assign the maximal weight to most preferred trajectories. In what follows, we discuss a class of weight functions that satisfy this property.

Definition 14
Let Ψ be a general preference. A weight function w_Ψ is called *admissible* if it satisfies the following properties:

(i) if $\alpha \prec_\Psi \beta$ then $w_\Psi(\alpha) > w_\Psi(\beta)$; and
(ii) if $\alpha \approx_\Psi \beta$ then $w_\Psi(\alpha) = w_\Psi(\beta)$.

It is easy to see that if w_Ψ is admissible then the following theorem will hold.

Proposition 6
Let Ψ be a general preference formula and $w_\Psi(\alpha)$ be an admissible weight function. If α is a trajectory such that $w_\Psi(\alpha)$ is maximal, then α is a most preferred trajectory w.r.t. Ψ.

Proof
Since $w_\Psi(\alpha)$ is maximal and w_Ψ is admissible, we conclude that there exists no trajectory β such that $\beta \prec_\Psi \alpha$. Thus, α is a most preferred trajectory w.r.t. Ψ. □

The above proposition implies that we can compute a most preferred trajectory using **smodels** if we can implement an admissible weight function. This is the topic of the next section. We would like to emphasize that the above result states *soundness* of this method, but *not completeness*. This means that the computation scheme proposed in the next section will return *a* most preferred trajectory, if the planning problem admits solutions. This is consistent and in line with the practice used in many well-known planning systems, in which only one solution is returned.

4.3 Computing an admissible weight function

Let Ψ be a general preference. We will now show how an admissible weight function w_Ψ can be built in a bottom-up fashion. We begin with the basic desires.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Definition 15 (*Basic Desire Weight*)

Let φ be a basic desire formula and let α be a trajectory. The weight of the trajectory α w.r.t. the basic desire φ is a function defined as

$$w_\varphi(\alpha) = \begin{cases} 1 & \text{if } \alpha \models \varphi \\ 0 & \text{otherwise} \end{cases}$$

The following proposition derives directly from the definition of admissibility.

Proposition 7

Let φ be a basic desire. Then w_φ is an admissible weight function.

The weight function of an atomic preference builds on the weight function of the basic desires occurring in the preference as follows.

Definition 16 (*Atomic Preference Weight*)

Let $\psi = \varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_k$ be an atomic preference formula. The weight of a trajectory α w.r.t. ψ is defined as follows:

$$w_\psi(\alpha) = \sum_{r=1}^{k} (2^{k-r} \times w_{\varphi_r}(\alpha))$$

Proposition 8

Let $\psi = \varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_k$ be an atomic preference formula, and let α_1, α_2 be two trajectories. Then

$$\alpha_1 \prec_\psi \alpha_2 \quad \textit{iff} \quad w_\psi(\alpha_1) > w_\psi(\alpha_2)$$

Furthermore, we also have that

$$\alpha_1 \approx_\psi \alpha_2 \quad \textit{iff} \quad w_\psi(\alpha_1) = w_\psi(\alpha_2).$$

Proof

Let us start by assuming $\alpha_1 \prec_\psi \alpha_2$. According to the definition, this means that $\exists (1 \leqslant i \leqslant k)$ such that

- $\forall (1 \leqslant j < i)(\alpha_1 \approx_{\varphi_j} \alpha_2)$
- $\alpha_1 \prec_{\varphi_i} \alpha_2$

From Proposition 7 we have that $\alpha_1 \approx_{\varphi_j} \alpha_2$ implies $w_{\varphi_j}(\alpha_1) = w_{\varphi_j}(\alpha_2)$ for $1 \leqslant j < i$. This leads to

$$\sum_{r=1}^{i-1} (2^{k-r} \times w_{\varphi_r}(\alpha_1)) = \sum_{r=1}^{i-1} (2^{k-r} \times w_{\varphi_r}(\alpha_2))$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

In addition, since $\alpha_1 \prec_{\varphi_i} \alpha_2$, then we also have that $1 = w_{\varphi_i}(\alpha_1) > w_{\varphi_i}(\alpha_2) = 0$. Thus, we have

$$\begin{aligned}
&\sum_{r=1}^{k}(2^{k-r} \times w_{\varphi_r}(\alpha_1)) &&= \\
&\sum_{r=1}^{i-1}(2^{k-r} \times w_{\varphi_r}(\alpha_1)) + 2^{k-i} + \sum_{r=i+1}^{k}(2^{k-r} \times w_{\varphi_r}(\alpha_1)) &&> \\
&\sum_{r=1}^{i-1}(2^{k-r} \times w_{\varphi_r}(\alpha_1)) + 2^{k-i} - 1 &&\geqslant \\
&\sum_{r=1}^{i-1}(2^{k-r} \times w_{\varphi_r}(\alpha_1)) + \sum_{r=i+1}^{k}(2^{k-r} \times w_{\varphi_r}(\alpha_2)) &&= \\
&\sum_{r=1}^{k}(2^{k-r} \times w_{\varphi_r}(\alpha_2))
\end{aligned}$$

For similar reasons, it is also easy to see that $\alpha_1 \approx_\psi \alpha_2$ implies $w_\psi(\alpha_1) = w_\psi(\alpha_2)$.

Let us now explore the opposite direction. Let us assume that $w_\psi(\alpha_1) > w_\psi(\alpha_2)$. It is easy to see that there must be an integer i, $1 \leqslant i \leqslant k$, such that $\alpha_1 \not\approx_{\varphi_i} \alpha_2$. Consider the minimal integer i satisfying this property, i.e., $\alpha_1 \approx_{\varphi_j} \alpha_2$ for every j, $1 \leqslant j < i$. Since $w_\psi(\alpha_1) > w_\psi(\alpha_2)$ and $\alpha_1 \not\approx_{\varphi_i} \alpha_2$ we conclude that $\alpha_1 \prec_{\varphi_j} \alpha_2$. This implies that $\alpha_1 \prec_\psi \alpha_2$.

Similar arguments allow us to prove that if $w_\psi(\alpha_1) = w_\psi(\alpha_2)$ then $\alpha_1 \approx_\psi \alpha_2$. □

We are now ready to define an admissible weight function w.r.t. a general preference.

Definition 17 (*General Preference Weight*)
Let Ψ be a general preference formula. The weight of a trajectory α w.r.t. Ψ (denoted by $w_\Psi(\alpha)$) is defined as follows:

- if Ψ is an atomic preference then the weight is defined as in Definition 16.
- if $\Psi = \Psi_1 \& \Psi_2$ then $w_\Psi(\alpha) = w_{\Psi_1}(\alpha) + w_{\Psi_2}(\alpha)$
- if $\Psi = \Psi_1 \mid \Psi_2$ then $w_\Psi(\alpha) = w_{\Psi_1}(\alpha) + w_{\Psi_2}(\alpha)$
- if $\Psi = \ !\Psi_1$ then $w_\Psi(\alpha) = max(\Psi_1) - w_{\Psi_1}(\alpha)$ where $max(\Psi_1)$ represents the maximum weight that a trajectory can achieve on the preference formulae Ψ_1 plus one.
- if $\Psi = \Psi_1 \lhd \Psi_2$[10] then $w_\Psi(\alpha) = max(\Psi_2) \times w_{\Psi_1}(\alpha) + w_{\Psi_2}(\alpha)$

We prove the admissibility of w_Ψ in the next proposition.

Proposition 9
For a general preference Ψ, w_Ψ is an admissible weight function.

The proof of this proposition is based on Propositions 7-8 and the structure of Ψ. It is omitted here for brevity. The above propositions allow us to prove the following result.

[10] Because of Proposition 5, without loss of generality, we describe the encoding only for chains of length 2.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Proposition 10
Let Ψ be a general preference and α be a trajectory with the maximal weight w.r.t. w_Ψ. Then, α is a most preferred trajectory w.r.t. Ψ.

Proof
Let $w_\Psi(\alpha)$ be maximal; let us assume by contradiction that there exists β such that $\beta \prec_\Psi \alpha$. It follows from the previous proposition that $\beta \prec_\Psi \alpha$ implies that $w_\Psi(\beta) < w_\Psi(\alpha)$, which contradicts the hypothesis that $w_\Psi(\alpha)$ is maximal. □

Propositions 7-9 show that we can compute an admissible weight function w_Ψ bottom-up from the weight of each basic desire occurring in Ψ. We are now ready to define the set of rules $\Pi_{pref}(\Psi)$, which consists of the rules encoding Ψ and the rules encoding the computation of w_Ψ. Similarly to the encoding of the basic desires, we will assign a new, distinguished name n_ϕ to each preference formula ϕ occurring in Ψ and encode the preferences in the same way as we did for the basic desires (Section 4.1). We will also add an atom $preference(n_\phi)$ to the set of atoms encoding ϕ. For brevity, we omit here the details of this step. The program $\Pi_{pref}(\Psi)$ defines two predicates, $w(p,n)$ and $max(p,n)$, where p is a preference name and n is the weight of the current trajectory with respect to the preference named p. $w(p,n)$ (resp. $max(p,n)$) is true if the weight (resp. maximal weight) of the current trajectory with respect to the preference p is n.

1. For each basic desire ϕ, $\Pi_{pref}(\phi)$ contains the rules encoding ϕ and the following rules:

$$\begin{array}{rcl} w(n_\phi,1) & \leftarrow & satisfy(n_\phi,0) \\ w(n_\phi,0) & \leftarrow & not\ satisfy(n_\phi,0) \\ max(n_\phi,2) & \leftarrow & \end{array} \tag{16}$$

2. For each atomic preference $\phi = \varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_k$, $\Pi_{pref}(\phi)$ consists of

$$\cup_{j=1}^{k} \Pi_{pref}(\varphi_k)$$

and the following rules:

$$\begin{array}{rcl} w(n_\phi,S) & \leftarrow & w(n_{\varphi_1},S_1),\ldots,w(n_{\varphi_k},S_k), S = \Sigma_{r=1}^{k} 2^{k-r} \times S_r \\ max(n_\phi,2^k) & \leftarrow & \end{array} \tag{17}$$

3. For each general preference Ψ,
   - if Ψ is an atomic preference then $\Pi_{pref}(\Psi)$ is defined as in the previous item.
   - if $\Psi = \Psi_1 \& \Psi_2$ or $\Psi = \Psi_1 | \Psi_2$ then $\Pi_{pref}(\Psi)$ consists of

$$\Pi_{pref}(\Psi_1) \cup \Pi_{pref}(\Psi_2)$$

and

$$\begin{array}{rcl} w(n_\Psi,S) & \leftarrow & w(n_{\Psi_1},N_1), w(n_{\Psi_2},N_2), S = N_1 + N_2 \\ max(n_\Psi,S) & \leftarrow & max(n_{\Psi_1},N_1), max(n_{\Psi_2},N_2), S = N_1 + N_2 \end{array} \tag{18}$$

   - if $\Psi = !\,\Psi_1$ then $\Pi_{pref}(\Psi)$ consists of $\Pi_{pref}(\Psi_1)$ and the rules

$$\begin{array}{rcl} w(n_\Psi,S) & \leftarrow & w(n_{\Psi_1},N), max(n_{\Psi_1},M), S = M - N \\ max(n_\Psi,S) & \leftarrow & max(n_{\Psi_1},S). \end{array} \tag{19}$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- if $\Psi = \Psi_1 \lhd \Psi_2$ then $\Pi_{pref}(\Psi)$ consists of $\Pi_{pref}(\Psi_1) \cup \Pi_{pref}(\Psi_2)$ and rules

$$\begin{array}{rcl} w(n_\Psi, S) & \leftarrow & w(n_{\Psi_1}, N_1), w(n_{\Psi_2}, N_2), \\ & & max(n_{\Psi_2}, M_2), S = M_2 * N_1 + N_2 \\ max(n_\Psi, S) & \leftarrow & max(n_{\Psi_1}, N_1), max(n_{\Psi_2}, N_2), \\ & & S = N_2 * N_1 + N_2 \end{array} \tag{20}$$

The next theorem proves the correctness of $\Pi_{pref}(\Psi)$.

Theorem 5
Let $\langle D, I, G\rangle$ be a planning problem, Ψ be a general preference, and $\alpha = s_0 a_1 \dots a_n s_n$ be a trajectory. Let $\Pi = \Pi_{pref}(\Psi) \cup \Pi_{sat} \cup \alpha^{-1}$. Then,

- For every desire ϕ with $desire(n_\phi) \in \Pi_{pref}(\Psi)$, we have that $\Pi \models w(n_\phi, w)$ iff $w_\phi(\alpha) = w$ and $\Pi \models max(n_\phi, w)$ iff $max(\phi) = w$.
- For every preference η with $preference(n_\eta) \in \Pi_{pref}(\Psi)$, we have that $\Pi \models w(n_\eta, w)$ iff $w_\eta(\alpha) = w$ and $\Pi \models max(n_\eta, w)$ iff $max(\eta) = w$

where

$$\alpha^{-1} = \{occ(a_i, i-1) \mid i \in \{1, \dots, n\}\} \cup \{holds(f, i) \mid f \in s_i, i \in \{0, \dots, n\}\}.$$

Proof
Let Π_1 be the program consisting of the rules (16)-(20) of the program $\Pi_{pref}(\Psi)$ and the set of atoms of the form $preference(n_\phi)$ in $\Pi_{pref}(\Psi)$. Let S be the set of literals occurring in the program $\Pi \setminus \Pi_1$. It is easy to check that S is a splitting set of Π. Using the Splitting Theorem (Lifschitz and Turner 1994), we can show that M is an answer set of Π iff $M = X \cup Y$, where X is an answer set of the program $\Pi \setminus \Pi_1$ and Y is an answer set of the program Π_2, which is obtained from Π_1 by replacing the rules of the form (16) with the set of atoms Z where

$$\begin{array}{rll} Z = \bigcup_{desire(n_\phi) \in X} & \{w(n_\phi, 1) \mid satisfy(n_\phi, 0) \in X\} & \cup \\ & \{w(n_\phi, 0) \mid satisfy(n_\phi, 0) \notin X\} & \cup \\ & \{max(n_\phi, 2)\}. & \end{array} \tag{21}$$

Observe that for a desire ϕ with $desire(n_\phi) \in \Pi_{pref}(\Psi)$ we have that $\Pi_\phi \subseteq \Pi_{pref}(\Psi)$. By applying the results of Theorem 2, we have that $\Pi \setminus \Pi_1$ has a unique answer set X and $satisfy(n_\phi, 0) \in X$ iff $\alpha \models \phi$. Together with the fact that $Z \subseteq Y$, we have that $w(n_\phi, 1) \in M$ iff $\alpha \models \phi$ iff $w_\phi(\alpha) = 1$ and $w(n_\phi, 0) \in M$ iff $\alpha \not\models \phi$ iff $w_\phi(\alpha) = 0$. Furthermore, $max(n_\phi, 2) \in M$ and $w_\phi(\alpha) \leqslant 1$ for every desire ϕ. This proves the first item of the theorem.

We will now prove the second item of the theorem. To account for the structure of the preference, we associate an integer, denoted by $\lambda(n_\phi)$, to each constant n_ϕ such that $preference(n_\phi) \in \Pi_{pref}(\Psi)$ or $desire(n_\phi) \in \Pi_{pref}(\Psi)$. This is done as follows:

- $\lambda(n_\phi) = 0$ if $desire(n_\phi) \in \Pi_{pref}(\Psi)$ (i.e., if ϕ is a desire);
- $\lambda(n_\phi) = 1$ if $preference(n_\phi) \in \Pi_{pref}(\Psi)$ and $\phi = \varphi_1 \lhd \varphi_2 \lhd \dots \lhd \varphi_k$;
- $\lambda(n_\phi) = \lambda(n_{\phi_1}) + \lambda(n_{\phi_2}) + 1$ if $preference(n_\phi) \in \Pi_{pref}(\Psi)$ and $\phi = \phi_1 \& \phi_2$ or $\phi = \phi_1 \mid \phi_1$; and
- $\lambda(n_\phi) = \lambda(n_{\phi_1}) + 1$ if $preference(n_\phi) \in \Pi_{pref}(\Psi)$ and $\phi = !\phi_1$.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

The proof is done inductively over $\lambda(n_\phi)$.

- **Base:** $\lambda(n_\phi) = 0$ means that n_ϕ is a desire. The claim for this case follows from the first item.
- **Step:** Assume that we have proved the conclusion for $\lambda(n_\phi) < k$. We will now prove it for $\lambda(n_\phi) = k$. Consider a preference ϕ with $\lambda(n_\phi) = k$. We have the following cases:
  - $\phi = \varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_k$ and φ_i are basic desires, i.e., ϕ is an atomic preference. By definition, we have that φ_i's are desires. It follows from (17) that
    $$w(n_\phi, s) \in Y \text{ iff the body of the first rule in (17) is satisfied by } Y$$
    $$\text{iff } s = \Sigma_{r=1}^{l} 2^{k-r} \times w_r \text{ and } w(n_{\varphi_i}, w_r) \in Z \text{ for } 1 \leqslant i \leqslant k$$
    $$\text{iff } s = w_\phi(\alpha).$$
    Furthermore, $max(n_\phi, 2^k) \in Y$ and the maximal weight of $w_\phi(\alpha)$ is $\Sigma_{r=1}^{k} 2^{k-r} = 2^k - 1$. This proves the inductive step for this case.
  - $\phi = \phi_1 \& \phi_2$ (resp. $\phi = \phi_1 \mid \phi_2$). We have that $\lambda(n_{\phi_1}) < k$ and $\lambda(n_{\phi_2}) < k$. The conclusion follows immediately from the induction hypothesis, the rules in (18), and the definition of $w_\phi(\alpha)$.
  - $\phi = !\phi_1$. Again, we have that $\lambda(n_{\phi_1}) < k$. Using (19) and the definition of w_ϕ, we can prove that $w(n_\phi, s) \in Y$ iff $w_\phi(\alpha) = s$ and $max(n_\phi, s) \in Y$ iff $max(\phi) = s$
  - $\phi = \phi_1 \lhd \phi_2$. Again, we have that $\lambda(n_{\phi_1}) < k$ and $\lambda(n_{\phi_2}) < k$. The conclusion follows immediately from (20), the induction hypothesis, and the definition of $w_\phi(\alpha)$.

□

The above theorem implies that we can compute a most preferred trajectory by (i) adding $\Pi_{pref}(\Psi) \cup \Pi_{sat}$ to $\Pi(D, I, G)$ and (ii) computing an answer set M in which $w(n_\Psi, w)$ is maximal. A working implementation of this is available in **jsmodels**.

4.4 Some examples of preferences in $\mathcal{PP}$

We will now present some preferences that are common to many planning problems and have been discussed in Eiter *et al.* (2003). The main difference between the encoding presented in this paper and the ones in (Eiter *et al.* 2003) lies in that we use temporal operators to represent the preferences, while action weights are used in Eiter *et al.* (2003). Let $\langle D, I, G \rangle$ be a planning problem. For the discussion in this subsection, we will assume that the answer set planning module $\Pi(D, I, G)$ is capable of generating trajectories without redundant actions in the sense that no action occurrence is generated once the goal has been achieved. Such a planning module can be easily obtained by adding a constraint to the program $\Pi(D, I, G)$ preventing action occurrences to be generated once the goal has been achieved. This, however, does not guarantee that the planning module will generate the shortest trajectory if n is greater than the length of the shortest trajectory. In keeping with the notation used in the previous section, we use φ to denote G (i.e., $\varphi = G$).

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

4.4.1 Preference for shortest trajectory – formula based encoding

Assume that we are interested in trajectories achieving φ whose length is less than or equal n. A simple encoding that allows us to accomplish such goal is to make use of basic desires. By $\mathbf{next}^i(\varphi)$ we denote the formula:

$$\underbrace{\mathbf{next}(\mathbf{next}(\mathbf{next}\cdots(\mathbf{next}}_{i}(\varphi))\cdots)).$$

Let us define the formula $\sigma^i(\varphi)$ $(0 \leqslant i \leqslant n)$ as follows:

$$\sigma^0(\varphi) = \varphi \qquad \sigma^i(\varphi) = \bigwedge_{j=0}^{i-1} \neg\mathbf{next}^j(\varphi) \ \wedge \ \mathbf{next}^i(\varphi)$$

Finally, let us consider the formula $short(n, \varphi)$ defined as

$$short(n, \varphi) = \sigma^0(\varphi) \lhd \sigma^1(\varphi) \lhd \sigma^2(\varphi) \lhd \cdots \lhd \sigma^n(\varphi).$$

Intuitively, this formula says that we prefer trajectories on which the goal φ is satisfied as early as possible. It is easy to see that if α is a most preferred trajectory w.r.t. $short(n, \varphi)$ then α is a shortest length trajectory satisfying the goal φ.

4.4.2 Preference for shortest trajectory – action based encoding

The formula based encoding $short(n, \varphi)$ requires the bound n to be given. We now present another encoding that does not require this condition. We introduce two additional fictitious actions *stop* and *noop* and a new fluent *ended*. The action *stop* will be triggered when the goal is achieved; *noop* is used to fill the slot so that we can compare between trajectories; the fluent *ended* will denote the fact that the goal has been achieved. We add to the action theory the propositions:

stop **causes** *ended*
stop **executable if** φ
noop **causes** *ended*
noop **executable if** *ended*

Furthermore, we add the condition $\neg$*ended* to the executability condition of every action in (D, I) and to the initial state I. We can encode the condition of shortest length trajectory as follows. Let

$$short = \mathbf{always}((stop \vee noop) <^e (a_1 \vee \ldots \vee a_k)).$$

where $a_1, \ldots, a_k$ are the actions in the original action theory. Again, we can show that any most preferred trajectory w.r.t. *short* is a shortest length trajectory satisfying the goal φ. Observe the difference between $short(n, \varphi)$ and *short*: both are built using temporal connectives but the former uses fluent formula and the latter uses actions. The second one, we believe, is simpler than the first one; however, it requires some modifications to the original action theory.

4.4.3 Cheapest plan

Let us assume that we would like to associate a cost $c(a)$ to each action a and determine trajectories that have the minimal cost. Since our comparison is done

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

only on trajectories whose length is less than or equal *length*, we will also introduce the two actions *noop* and *stop* with no cost and the fluent *ended* to record the fact that the goal has been achieved. Furthermore, we introduce the fluent $sCost(ct)$ to denote the cost of the trajectory. Intuitively, $scost(ct)$ is true mean that the cost of the trajectory is ct. Initially, we set the value of $sCost$ to 0 (i.e., $sCost(0)$ is true initially and $sCost(c)$ is false for every other c) and the execution of action a will increase the value of $sCost$ by $c(a)$. This is done by introducing an effect proposition

$$a \textbf{ causes } sCost(N + c(a)) \textbf{ if } sCost(N)$$

for each action a[11]. The preference

$$\textbf{goal}(sCost(m)) \lhd \textbf{goal}(sCost(m+1)) \ldots \lhd \textbf{goal}(sCost(M))$$

where m and M are the estimated minimal and maximal cost of the trajectories, respectively. Note that we can have $m = 0$ and $M = max\{c(a) \mid a \text{ is an action}\} \times length$.

5 Related work

The work presented in this paper is the natural continuation of the work we presented in Son and Pontelli (2004a), where we rely on prioritized default theories to express limited classes of preferences between trajectories – a strict subset of the preferences covered in this paper. This work is also influenced by other works on exploiting *domain-specific knowledge* in planning (Bacchus and Kabanza 2000; Dal Lago, Pistore, and Traverso 2002; Son *et al.* 2005), in which domain-specific knowledge is expressed as a constraint on the trajectories achieving the goal, and hence, is a *hard constraint*. In Section 5.1, we discuss different approaches to planning with preferences which are directly related to our work. In Sections 5.2–5.3 we present works that are somewhat related to our work and can be used to develop alternative implementation for $\mathscr{PP}$.

5.1 Planning with preferences

Different approaches have been proposed to integrate preferences in the planning process. An approach close in spirit to the one proposed in this paper has been recently developed by Delgrande *et al.* (2004). The framework they propose introduces qualitative preferences built from two partial preorders, $\leqslant_c$ and $\leqslant_t$, over the set of propositional formulae of fluents and actions. Intuitively,

- $\varphi_1 \leqslant_c \varphi_2$ (*choice order*) indicates the desire to prefer trajectories that satisfy (at some point in time) the formula φ_2 over those that satisfy φ_1.
- $\varphi_1 \leqslant_t \varphi_2$ (*temporal order*) indicates the desire to prefer trajectories that satisfy φ_1 first and φ_2 later in the trajectory.

[11] Because of the grounding requirement of answer set solver, this encoding will yield a set of effect propositions instead of a single proposition.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Choice preferences are employed to derive an ordering $\lhd_c$ between trajectories as follows: given trajectories α, β we have that

$$\alpha \lhd_c \beta \ \ iff \ \forall \varphi \in \Delta(\alpha, \beta). \exists \varphi' \in \Delta(\beta, \alpha). (\varphi \leqslant_c \varphi')$$

where $\Delta(\gamma, \gamma') = \{\varphi \in dom(\leqslant_c) \mid \gamma \models \varphi, \gamma' \not\models \varphi\}$ and $\gamma \models \varphi$ denotes the fact that the formula φ is true at one of the states reached by the trajectory γ. The order is made transitive by taking the transitive closure of $\lhd_c$.

The relation $\lhd_c$ can be easily simulated in $\mathscr{PP}$ since $\varphi \leqslant_c \varphi'$ determines the same order as **eventually**$(\varphi') \lhd$ **eventually**(φ). This can be generalized as long as $\leqslant_c$ is cycle-free.

Example 8

Let us consider the monkey-and-banana example as formulated in Delgrande *et al.* (2004). The world includes the following entities: a monkey, a banana hanging from the ceiling, a coconut on the floor, and a chocolate bar inside a closed drawer. Initially, all the entities are in different locations in a room. The room includes also a box that can be pushed and climbed on to reach the ceiling and grab the banana. The goal is to get the chocolate as well as at least one of the banana or the coconut. The domain description includes the following fluents:

- *location*$(Entity, Location)$ denoting the current *Location* of *Entity*; the domain of *Entity* is $\{monkey, banana, coconut, drawer, box\}$ and the domain of *Location* is $\{1, \ldots, 5\}$ (denoting 5 different positions in the room).
- *onBox* denoting the fact that the monkey is on top of the box.
- *hasBanana* denoting the fact that the monkey has the banana.
- *hasCoconut* denoting the fact that the monkey has the coconut.
- *hasChocolate* denoting the fact that the monkey has the chocolate.
- *DrawerOpen* denoting the fact that the drawer is open.

The action theory provides actions to walk in the room, move the box, climb on and off the box, grab objects, and open drawers. The goal considered here is expressed by the fluent formula:

$$hasChocolate \wedge (hasCoconut \vee hasBanana)$$

The preference discussed in Delgrande *et al.* Delgrande *et al.* (2004) is that bananas are preferred over coconuts – i.e., $hasCoconut \leqslant_c hasBanana$ – and in our framework it can be expressed as

$$\textbf{eventually}(hasBanana) \lhd \textbf{eventually}(hasCoconut).$$

This preference can also be represented by a simpler basic desire

$$\textbf{goal}(hasBanana)$$

which says that trajectories achieving *hasBanana* will be most preferred. □

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Temporal preferences are employed to derive another preorder $\lhd_t$ between trajectories as follows:

- given a trajectory $\alpha = s_0 a_1 \dots a_n s_n$ and two propositions over fluent and actions φ, φ', then $\varphi \leqslant_\alpha \varphi'$ iff
  — $\alpha \models \varphi$ and $\alpha \models \varphi'$ and
  — $i_\varphi \leqslant i_{\varphi'}$ where s_{i_φ} (resp. $s_{i_{\varphi'}}$) is the first state in α that satisfies φ (resp. φ').
- given two trajectories α, β, we have that $\alpha \lhd_t \beta$ iff $<_t \cap \leqslant_\beta^{-1} \subseteq <_t \cap \leqslant_\alpha^{-1}$ where $\leqslant_\alpha^{-1}$ is the inverse relation of $\leqslant_\alpha$.

Each individual temporal preference $\varphi \leqslant_t \varphi'$ can be expressed in our language as the basic desire

$$c(\varphi \leqslant_t \varphi') \equiv \textbf{eventually}(\varphi \wedge \textbf{eventually}(\varphi')) \wedge \textbf{until}(\neg\varphi', \varphi)$$

The generalization to a collection of temporal preferences requires some additional constructions. Given a collection of basic desires $S = \{\psi_1, \dots, \psi_k\}$, then

- for an arbitrary permutation $i_1, \dots, i_k$ of $1, \dots, k$, let us define

$$ch(S, i_1, \dots, i_k) \equiv \bigwedge_{j=1}^{k} \psi_{i_j} \lhd \bigwedge_{j=2}^{k} \psi_{i_j} \lhd \bigwedge_{j=3}^{k} \psi_{i_j} \lhd \cdots \lhd \psi_{i_k}.$$

  Intuitively, $ch(S, i_1, \dots, i_k)$ is an atomic preference representing an ordering between trajectories w.r.t. the set of basic desires $\{p_{i_1}, \dots, p_{i_k}\}$. For example, trajectories satisfying $\bigwedge_{j=1}^{k} \psi_{i_j}$ is most preferred; if no trajectory satisfies $\bigwedge_{j=1}^{k} \psi_{i_j}$ then trajectories satisfying $\bigwedge_{j=2}^{k} \psi_{i_j}$ is most preferred; etc.
- let $\{\pi_1, \dots, \pi_{k!}\}$ be the set of all permutations of $1, \dots, k$; let us define

$$maxim(S) \equiv ch(S, \pi_1) \mid ch(S, \pi_2) \mid \cdots \mid ch(S, \pi_{k!}).$$

  Intuitively, $maxim(S)$ indicates that we prefer trajectories satisfying the maximal number of basic desires from the set $\{\psi_1, \dots, \psi_k\}$.

If we have a collection of temporal preferences $\{\varphi_i \leqslant_t \varphi_i' \mid i = 1, \dots, k\}$, then the equivalent formula is

$$maxim(\{c(\varphi_i \leqslant_t \varphi_i') \mid i = 1, \dots, k\}).$$

Example 9

Let us continue Example 8 by removing the choice preference and assuming instead the temporal preference $hasBanana \leqslant_t hasChocolate$ – i.e., the banana should be obtained before the chocolate. The corresponding encoding in our language is

$$\textbf{eventually}(hasBanana \wedge \textbf{eventually}(hasChocolate)) \quad \wedge \\ \textbf{until}(\neg hasChocolate, hasBanana)$$

□

Eiter *et al.* (2003) introduced a framework for planning with action costs using logic programming. The focus of their proposal is to express certain classes of quantitative preferences. Each action is assigned an integer cost, and plans with the minimal cost are considered to be optimal. Costs can be either static or relative to the time step in which the action is executed. Eiter *et al.* (2003) also present

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

the encoding of different preferences, such as shortest plan and the cheapest plan. Our approach also emphasizes the use of logic programming, but differs in several aspects. Here, we develop a *declarative language* for preference representation. Our language can express the preferences discussed in Eiter *et al.* (2003), but it is more high-level and flexible than the action costs approach. The approach in Eiter *et al.* (2003) also does not allow the use of fully general dynamic preferences. On the other hand, while we only consider planning with complete information, Eiter *et al.* (2003) deal with planning in the presence of incomplete information and non-deterministic actions.

Other systems have adopted fixed types of preferences, e.g., shortest plans (Cimatti and Roveri 2000; Blum and Furst 1997).

Our proposal has similarities with the approach based on metatheories of the planning domain (Myers 1996; Myers and Lee 1999), where metatheories provide characterization of semantic differences between the various domain operators and planning variables; metatheories allow the generation of biases to focus the planner towards plans with certain characteristics.

Our work is also related to the work in Lin (1998) in which the author defined three different measures for plan quality (A-, B-, and C-optimal) and showed how they can be axiomatized in situation calculus. Roughly, a plan is A-optimal if none of its actions can be deleted and the remainder is still a valid plan. It is B-optimal if none of its segments can be deleted and the remainder is still a plan. It is C-optimal if none of its segments can be replaced by a single action and the remainder is still a plan. While these measures are domain-independent, preferences in our language are mostly domain-dependent. Theoretically, these measures could also be expressed in $\mathscr{PP}$ by defining an order among possible plans. This impractical method can be replaced by considering some approximations of these measures. As an example, shortest plans as encoded in the previous section represents a class of A-optimal plans; the atomic preference $\varphi_1 \lhd \varphi_2$ with $\varphi_1 = occ(a) \wedge \mathbf{executable}(a)$ and $\varphi_2 = occ(b) \wedge \mathbf{executable}(b) \wedge \mathbf{next}(occ(c) \wedge \mathbf{executable}(c))$ could be used to prefer plans with action a over plan containing the sequence $b; c$; etc.

Our language allows the representation of several types of preferences similar to those developed in Haddawy and Hanks (1993) for decision-theoretic planners. The fundamental difference is that we use logic programming while their system is probability based. Our approach also differs from the works on using Markov Decision Processes (MDP) to find optimal plans (Putterman 1994); in MDPs, optimal plans are functions from states to actions, thus preventing the user from selecting preferred trajectories without changing the MDP specification.

5.2 High-level languages for qualitative preferences

Brewka recently proposed (Brewka 2004a) a general rank-based description language for the representation of qualitative preferences between models of a propositional theory. The language has similar foundations to our proposal. The basic preference between models derives from an inherent total preorder between propositional formulae (*Ranked Knowledge Base*); models can be compared according to

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

one of four possible comparison criteria – i.e., *inclusion*, *cardinality*, *maximal degree of satisfied formula*, and *maximal degree of unsatisfied formula*. The preference language allows the refinement of the basic preference by using propositional combination as well as meta-ordering between preferences, in a fashion similar to what described in this paper.

The proposal by Junker (2001) presents a language designed to express preferences between decisions and decision rules in the context of a language for solving configuration problems. Decisions are described by labeled constraints $t : \varphi$, where t is a term (possibly containing variables) and φ is a configuration constraint. The configuration language allows also the creation of named sets of decisions. Preferences between decisions are expressed through statements of the form $prefer(t_1, t_2)$, where t_1, t_2 identify decisions or sets of decisions. The language allows the user also to create constraints that assert decisions, thus making it possible to express meta-preferences. For example, the following decisions express different preferences (Junker 2001):

```
decision rule p1(x):
   if x in instances(Customer) and playboy in characteristics(x)
        then prefer(look, comfort)
decision rule p2(x):
   if x in instances(Customer) and age(x)=old
        then prefer(comfort,look)
```

and a statement of the type $prefer(p1, p2)$ allows to express a meta-preference.

5.3 Other related works

Considerable effort has been invested in developing frameworks for expressing preferences within the context of constraint programming and constraint logic programming, where the problem of inconsistency arises frequently. Most proposals rely on the idea of associating preferences (expressed as mathematical entities) to constraints when variables are assigned (Schiex and Cooper 2002). Combinations of constraints lead to corresponding combinations of preferences, and the frameworks provide means to compare preferences; comparisons are commonly employed to select solutions or to discriminate between classes of satisfied constraints. A popular scheme relies on the use of *costs* associated to tuples (where each tuple represent a value assignment), and costs are drawn from a semiring structure (Schiex *et al.* 1995; Bistarelli *et al.* 1997), which provides operators to combine preferences and to "maximize" preferences. These frameworks subsume various approaches to preferences in CSP (Borning *et al.* 1989; Moulin 1988; Fargier and Lang 1993).

Larrosa and Schiex (2003) and Schiex and Cooper (2002) recently proposed the notion of *Valued CSP* as an algebraic framework for preferences in constraint network. In VCSP, costs for tuples are drawn from a value structure $\langle E, \oplus, \succeq \rangle$, where E is totally ordered by $\succeq$; the maximum denotes total inconsistency. Intuitively, in a valued CSP, each constraint c_X over a set of variables X is viewed as a function that maps tuples of values (values drawn from the domains of the variables X) to an

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

element of E (the "cost" of the tuple). The cost of the constraint allows us to rank the "degree" of constraint violation. Given an assignment t for a set of constraints C, the valuation of the assignment is the $\oplus$ composition of the E values of the individual constraints. The objective is to determine an assignment which is minimal w.r.t. the order $\succeq$. *Weighted CSP (WCSP)* are instances of this framework, where $E = [0, 1, \ldots, k]$, $\succeq$ is the standard ordering between natural numbers, and $\oplus$ is defined as $a \oplus b = min\{k, a + b\}$. Extensions of arc-consistency to these frameworks have been investigated (Larrosa 2002; Larrosa and Schiex 2003; Bistarelli *et al.* 1997).

Qualitative measures of preference in constraint programming have been explored through the notion of *Ceteris Paribus Networks (CP-nets)* (Boutilier *et al.* 1999).

A CP-net is a graphical tool to represent qualitative preferences. Let $\mathscr{V}$ be a set of variables and let us denote with $D(v)$ the domain of variable v ($v \in \mathscr{V}$). A CP-net is a pair $\langle G, P \rangle$, where G is a directed (typically acyclic) graph whose vertices are elements of $\mathscr{V}$, while the edges denote preferential dependences between variables; intuitively, preferences for a value for a variable v depend only on the values selected for the parents of v in the network. For a given assignment of values to the parents of v, the CP-net specifies a total order on $D(v)$. An assignment of values γ to $\mathscr{V}$ is immediately preferred to the assignment η if there is a variable v such that

- $\forall u \in \mathscr{V} \setminus \{v\}. \gamma(u) = \eta(u)$
- $\gamma(v)$ is preferred to $\eta(v)$ in the ordering of $D(v)$ specified by the assignment γ to the parents of v.

In general, an assignment γ is CP-preferred to an assignment η if there exists a sequence of assignments $\gamma_0, \gamma_1, \ldots, \gamma_k$ such that

- $\gamma_0 = \eta$
- γ_i is immediately preferred to γ_{i-1}
- $\gamma_k = \gamma$

Algorithms for solving constraint optimization problems under preference ordering specified by CP-nets have been proposed in the literature (Domshlak and Brafman 2002; Boutilier *et al.* 2004).

Constraint solving has also been proposed for the management of planning in presence of action costs (Kautz and Walser 1999).

Considerable effort has been invested in introducing preferences in logic programming. In Cui and Swift (2002), preferences are expressed at the level of atoms and used for parsing disambiguation in logic grammars. Rule-level preferences have been used in various proposals for selection of preferred answer sets in answer set programming (Brewka and Eiter 1999; Delgrande *et al.* 2003; Gelfond and Lifschitz 1998; Schaub and Wang 2001). Some of the existing answer set solvers include limited forms of (numerical) optimization capabilities. **smodels** (Simons *et al.* 2002) offers the ability to associate *weights* to atoms and to compute answer sets that minimize or maximize the total weight. DLV (Buccafurri *et al.* 2000) provides the notion of *weak constraints*, i.e., constraints of the form

$$\leftarrow \ell_1, \ldots, \ell_k. [w : l]$$

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

where w is a numeric penalty for violating the constraint, and l is a priority level. The total cost of violating constraints at each priority level is computed, and answer sets are compared to minimize total penalty (according to a lexicographic ordering based on priority levels).

5.4 Alternative encodings of $\mathscr{PP}$

In this section we discuss the possibility of implementing $\mathscr{PP}$ using inference backends different from **smodels** or **jsmodels**. It should be noted that the encoding proposed in Section 4 can be translated into **dlv** code with little effort, while it is not so with other answer set programming systems (e.g., **cmodels**, **ASSAT**), since they do not offer a construct similar to the **maximize** construct of **smodels**.

In this section, we explore the relationships between $\mathscr{PP}$ and two relatively new answer set programming frameworks, *Logic Programming with Ordered Disjunctions* and *Answer Set Optimizations*. The key idea in both cases, is to show that each preference of $\mathscr{PP}$ can be mapped to a collection of rules in these two languages. Below we provide some details of these languages and their use for expressing $\mathscr{PP}$.

5.4.1 Logic Programming with Ordered Disjunctions (LPOD)

Overview of LPOD: In *Logic Programming with Ordered Disjunctions* (Brewka *et al.* 2002), a program is a collection of ground rules of the form

$$A_1 \times \cdots \times A_k \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$$

The literals in the head of the rule represent alternative choices; in the specific case of LPOD, the choices are ordered, where A_1 is the most preferred choice, while A_k is the least preferred one.

The semantics of a LPOD program P is based on the general idea of answer sets and the concept of *split* of a program. For each rule $A_1 \times \cdots \times A_k \leftarrow Body$, the i^{th} option of the rule is the standard logic programming clause

$$A_i \leftarrow Body, not\ A_1, \ldots, not\ A_{i-1}$$

A split of the program P is a standard logic program obtained by replacing each rule of P by one of its options.

Given a LPOD program P and a set of ground literals S, then S is an answer set of P iff S is an answer set of a split of P.

Ordered disjunctions are employed to create a preference order between answer sets of a program. Different ordering criteria have been discussed (Brewka *et al.* 2002). Given an answer set S of a LPOD program P, we say that S satisfies a rule r

$$A_1 \times \cdots \times A_k \leftarrow Body$$

- with degree 1 ($deg_S(r) = 1$) if $S \not\models Body$
- with degree i ($deg_S(r) = i$) if $S \models Body$ and $i = min\{j \mid S \models A_j\}$

We denote with $S^i(P) = \{r \in P \mid deg_S(r) = i\}$. The three criteria for comparing

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

answer sets under LPOD are the following. Let S_1, S_2 be two answer sets of P;

- S_1 is cardinality preferred to S_2 ($S_1 >_c S_2$) iff $\exists i$ such that $|S_1^j(P)| = |S_2^j(P)|$ for $j < i$ and $|S_1^i(P)| > |S_2^i(P)|$.
- S_1 is inclusion preferred to S_2 ($S_1 >_i S_2$) iff $\exists i$ such that $S_1^j(P) = S_2^j(P)$ for $j < i$ and $S_1^i(P) \supset S_2^i(P)$.
- S_1 is Pareto preferred to S_2 ($S_1 >_p S_2$) iff
  - — $\exists r \in P.\, deg_{S_1}(r) < deg_{S_2}(r)$
  - — $\nexists r' \in P.\, deg_{S_1}(r) > deg_{S_2}(r)$

LPOD allows also the use of meta-preferences between rules of the form $r_1 \succ r_2$. The Pareto preference in this case is modified as follows: $S_1 >_p S_2$ iff

- $\exists r \in P.\, deg_{S_1}(r) < deg_{S_2}(r)$
- $\forall r \in P$, if $deg_{S_1}(r) > deg_{S_2}(r)$ then there exists another rule r' such that $r' \succ r$ and $deg_{S_1}(r') < deg_{S_2}(r)$

Translation of our Preferences: Let us start by providing a logic programming encoding of basic desires. We define two entities: $Core^{\psi}(T)$ is a unary predicate while $rules^{\psi}(T)$ is a collection of rules where T is a variable representing time step.

- if $\psi \equiv occ(a)$ ($a \in \mathbf{A}$) then $Core^{\psi}(T) = occ(a, T)$ and $rule^{\psi}(T) = \emptyset$.
- if $\psi \equiv f$ (f is a fluent literal) then $Core^{\psi}(T) = holds(f, T)$ and $rule^{\psi}(T) = \emptyset$.
- if $\psi \equiv \psi_1 \wedge \psi_2$ then $Core^{\psi}(T) = p^{\psi}(T)$ and
$$rule^{\psi}(T) = \left\{ \begin{array}{l} \leftarrow p^{\psi}(T), not\ Core^{\psi_1}(T) \\ \leftarrow p^{\psi}(T), not\ Core^{\psi_2}(T) \\ p^{\psi}(T) \leftarrow Core^{\psi_1}(T), Core^{\psi_2}(T) \end{array} \right\}$$
  where p^{ψ} is a new unary predicate.
- if $\psi \equiv \psi_1 \vee \psi_2$ then $Core^{\psi}(T) = p^{\psi}(T)$ and
$$rule^{\psi}(T) = \left\{ \begin{array}{l} \leftarrow p^{\psi}(T), not\ Core^{\psi_1}(T), not\ Core^{\psi_2}(T) \} \\ p^{\psi}(T) \leftarrow Core^{\psi_1}(T) \\ p^{\psi}(T) \leftarrow Core^{\psi_2}(T) \end{array} \right\}$$
  where p^{ψ} is a new unary predicate.
- if $\psi \equiv \neg\psi_1$ then $Core^{\psi}(T) = p^{\psi}(T)$ and
$$rule^{\psi}(T) = \left\{ \begin{array}{l} p^{\psi}(T) \leftarrow not\ Core^{\psi_1}(T) \\ \leftarrow p^{\psi}(T), Core^{\psi_1}(T) \end{array} \right\}$$
  where p^{ψ} is a new unary predicate.
- if $\psi \equiv \mathbf{next}(\psi_1)$ then $Core^{\psi}(T) = p^{\psi}(T)$ and
$$rule^{\psi}(T) = \left\{ \begin{array}{l} \leftarrow p^{\psi}(T), not\ Core^{\psi_1}(T+1) \\ p^{\psi}(T) \leftarrow Core^{\psi_1}(T+1) \end{array} \right\}$$
  where p^{ψ} is a new unary predicate.
- if $\psi \equiv \mathbf{always}(\psi_1)$ then $Core^{\psi}(T) = p^{\psi}(T)$ and
$$rule^{\psi}(T) = \left\{ \begin{array}{l} \leftarrow p^{\psi}(T), not\ Core^{\psi_1}(T1), T \leqslant T1, T1 \leqslant n \\ p^{\psi}(T) \leftarrow always^{\psi_1}(T) \\ always^{\psi_1}(n) \leftarrow Core^{\psi_1}(n) \\ always^{\psi_1}(T) \leftarrow T < n, Core^{\psi_1}(T), always^{\psi_1}(T+1) \end{array} \right\}$$
  where p^{ψ} is a new unary predicate.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

- if $\psi \equiv$ **eventually**(ψ_1) then $Core^{\psi}(T) = p^{\psi}(T)$ and

$$rule^{\psi}(T) = \left\{ \begin{array}{l} \leftarrow p^{\psi}(T), not\ Core^{\psi_1}(T), not\ Core^{\psi_1}(T+1), \ldots, Core^{\psi_1}(n) \\ p^{\psi}(T) \leftarrow Core^{\psi_1}(T1), T \leqslant T1, T1 \leqslant n \end{array} \right\}$$

  where p^{ψ} is a new unary predicate.

Let us define as $\Pi^{\psi}(i) = rule^{\psi}(i) \cup \{Core^{\psi}(i) \times \neg Core^{\psi}(i)\}$ and let us denote with $r^{\psi}(i)$ the rule $Core^{\psi}(i) \times \neg Core^{\psi}(i)$. We can show that if $\langle D, I, G \rangle$ is a planning problem and ψ is a basic desire, then the following holds:

$$S_1 >_p S_2 \quad iff \quad \pi(S_1) \models \psi \ and \ \pi(S_2) \not\models \psi$$

where S_1, S_2 are two answer sets of $\Pi(D, I, G) \cup \Pi^{\psi}(i)$ and $\pi(S)$ denotes the trajectory represented by S.

Let us extend the encoding above to include atomic preferences. In particular, given an atomic preference of the form $\psi_1 \lhd \psi_2$, we define

$$\Pi^{\psi}(i) = rule^{\psi_1}(i) \cup rule^{\psi_2}(i) \cup \left\{ \begin{array}{ll} (r1) & Core^{\psi_1}(i) \times not\ Core^{\psi_1}(i) \\ (r2) & Core^{\psi_2}(i) \times not\ Core^{\psi_2}(i) \\ r1 \succ r2 & \end{array} \right\}$$

A result similar to the one above can be derived: for a planning problem $\langle D, I, G \rangle$ and an atomic preference ψ, if S_1, S_2 are two answer sets of $\Pi(D, I, G) \cup \Pi^{\psi}(i)$, then

$$S_1 >_p S_2 \quad iff \quad \pi(S_1) \prec_{\psi} \pi(S_2).$$

The encoding of general preferences in the LPOD framework does not appear to be as simple as in the previous cases. The encoding is clearly possible – it is sufficient to make use of the encoding presented in Section 4; if ψ is the preference and $max(n_{\psi}, v)$ is true, then we can introduce the rule

$$w(n_{\psi}, v) \times w(n_{\psi}, v-1) \times \cdots \times w(n_{\psi}, 1).$$

The resulting encoding, on the other hand, is not any simpler than the direct encoding in **smodels** with atom weights.

5.4.2 Answer Set Optimization (ASO)

Overview of Answer Set Optimization: The paradigm of *Answer Set Optimization* was originally introduced by Brewka, Niemelä, and Truszczyński (Brewka *et al.* 2003) and later refined by Brewka (2004b).

In ASO, a program is composed of two parts $\langle P_{gen}, P_{pref} \rangle$, where P_{gen} is an arbitrary logic program (the *generator* program) and P_{pref} is a collection of preference rules, used to define a preorder over the answer sets of P_{gen}. The basic type of rules present in P_{pref} are of the form

$$C_1 : p_1 > \ldots > C_n : p_n \leftarrow Body \tag{22}$$

where p_i are numerical weights while C_j are propositional formulae. The complex types of preference rules in P_{pref} is defined inductively using rules of the form (22) and the constructors *psum, inc, rinc, card, rcard, pareto,* and *lex*.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

For each rule r of the form (22), an answer set S of the program $\langle P_{gen}, P_{pref} \rangle$ yields a penalty $pen(S,r)$ which is defined by (i) $pen(S,r) = p_j$ where $j = \min\{i \mid S \models C_i\}$ if S satisfies $Body$ and at least one C_i, and (ii) $pen(S,r) = 0$ otherwise. This penalty is used in defining a preorder among answer sets of the program as follows.

Given two answer sets S_1, S_2 of an ASO program P, we have that S_1 is preferred to S_2 ($S_1 \geqslant S_2$) w.r.t. a rule r in P of the form (22) if

$$pen(S_1, r) \leqslant pen(S_2, r).$$

More complex types of preorder can be described by combining preference rules using a predefined set of constructors:

- $(psum\ e_1, \ldots, e_k)$, where $S_1 \geqslant S_2$ iff

$$\sum_{i=1}^{k} pen(S_1, e_i) \leqslant \sum_{i=1}^{k} pen(S_2, e_i)$$

- $(rinc\ e_1, \ldots, e_k)$, where $S_1 \geqslant S_2$ iff

$$\exists 1 \leqslant i \leqslant k.(Pen^i(S_1) \supset Pen^i(S_2) \wedge \forall j < i.(Pen^j(S_1) = Pen^j(S_2)))$$

or

$$\forall 1 \leqslant i \leqslant k.(Pen^i(S_1) = Pen^i(S_2))$$

where $Pen^i(S) = \{j \mid pen(S, e_j) = i\}$.

- $(rcard\ e_1, \ldots, e_k)$, where $S_1 \geqslant S_2$ iff

$$\exists 1 \leqslant i \leqslant k.(|Pen^i(S_1)| > |Pen^i(S_2)| \wedge \forall j < i.(|Pen^j(S_1)| = |Pen^j(S_2)|))$$

or

$$\forall 1 \leqslant i \leqslant k.(|Pen^i(S_1)| = |Pen^i(S_2)|)$$

- $(lex\ e_1, \ldots, e_k)$, where $S_1 \geqslant S_2$ iff

$$\exists 1 \leqslant i \leqslant k.(S_1 >_i S_2 \wedge \forall j < i.(S_1 \geqslant_j S_2))$$

or

$$\forall 1 \leqslant i \leqslant k.(S_1 \geqslant_i S_2)$$

where $\geqslant_i$ is the preorder associated to the expression e_i.

- $(pareto\ e_1, \ldots, e_k)$, where $S_1 \geqslant S_2$ iff

$$\forall 1 \leqslant i \leqslant k.(S_1 \geqslant_i S_2)$$

where each e_i is a preference rule in P_{pref}.

Encoding of our Preferences: As for the case of LPOD, the encoding of our preference language in ASO is simple for the first two levels (basic desires and atomic preferences), while it is more complex in the case of general preferences.

We will follow an encoding structure that is analogous to the one used in Section 5.4.1. In particular, we maintain the same definition of $Core^{\psi}(T)$ and $rules^{\psi}(T)$. In this case, the generator program P_{gen} corresponds simply to the $\Pi(D, I, G)$ program that encodes the planning problem. The preference rules employed in the various

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

cases are the following:

- if $\psi(T) \equiv occ(a)$ then
$$e^{\psi}(T) \equiv occ(a, T) > \neg occ(a, T) \leftarrow .$$
- if $\psi(T) \equiv f$ (where f is a fluent literal) then
$$e^{\psi}(T) \equiv holds(a, T) > \neg holds(a, T) \leftarrow .$$
- if $\psi(T) \equiv \psi_1(T) \wedge \psi_2(T)$ then
$$e^{\psi}(T) \equiv (Core^{\psi_1}(T) \wedge Core^{\psi_2}(T)) > \top \leftarrow .$$
(where $\top$ is a tautology).
- if $\psi(T) \equiv \psi_1(T) \vee \psi_2(T)$ then
$$e^{\psi}(T) \equiv (Core^{\psi_1}(T) \vee Core^{\psi_2}(T)) > \top \leftarrow .$$
- if $\psi(T) \equiv \neg\psi_1(T)$ then
$$e^{\psi}(T) \equiv \neg Core^{\psi_1}(T) > Core^{\psi_1}(T) \leftarrow .$$
- if $\psi(T) \equiv \mathbf{next}(\psi_1(T))$ then
$$e^{\psi}(T) \equiv Core^{\psi_1}(T+1) > \neg Core^{\psi_1}(T+1) \leftarrow .$$
- if $\psi(t) \equiv \mathbf{eventually}(\psi_1(t))$ (with $1 \leqslant t \leqslant n$, where n is the length of the desired plan) then
$$e^{\psi}(t) \equiv (Core^{\psi_1}(t) \vee Core^{\psi_1}(t+1) \vee \ldots \vee Core^{\psi_1}(n)) > \top \leftarrow .$$
- if $\psi(t) \equiv \mathbf{always}(\psi_1(t))$ then
$$e^{\psi}(t) \equiv (Core^{\psi_1}(t) \wedge Core^{\psi_1}(t+1) \wedge \ldots \wedge Core^{\psi_1}(n)) > \top \leftarrow .$$

With respect to the original definition of ASO, which allows for a ranked sequence of preference programs, atomic preferences of the type $\psi_1 \lhd \psi_2$ can be encoded as $\langle\{e^{\psi_1}\}, \{e^{\psi_2}\}\rangle$. In the extended ASO model proposed in Brewka (2004b), the same effect can be obtained by using the expression

$$(pareto\ e^{\psi_1}, e^{\psi_2})$$

Only some of the general preferences can be directly encoded without relying on the use of numeric weights.

- if $\psi \equiv \psi_1 \& \psi_2$, then we can introduce the expression
$$e^{\psi} \equiv (pareto\ e^{\psi_1}, e^{\psi_2})$$
- if $\psi \equiv \psi_1 \lhd \psi_2$, then we can introduce the expression
$$e^{\psi} \equiv (lex\ e^{\psi_1}, e^{\psi_2})$$

The other cases appear to require the use of weights, leading to an encoding as complex as the one presented in Section 4.

For the cases listed above, we can assert the following result: for a planning problem $\langle D, I, G\rangle$, a preference ψ, and two answer sets S_1, S_2 of $\Pi(D, I, G)$, it holds

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

that

$$S_1 \geqslant_\psi S_2 \quad \textit{iff} \quad \pi(S_1) \preceq_\psi \pi(S_2)$$

where $\geqslant_\psi$ is the preorder derived from the expression e^ψ.

6 Conclusion and future work

In this paper we presented a novel declarative language, called $\mathscr{PP}$, for the specification of preferences in the context of planning problems. The language nicely integrates with traditional action description languages (e.g., $\mathscr{B}$) and it allows the elegant encoding of complex preferences between trajectories. The language provides a *declarative* framework for the encoding of preferences, allowing users to focus on the high-level description of preferences (more than their encodings – as in the approaches based on utility functions). $\mathscr{PP}$ allows the expression of complex preferences, including multi-dimensional preferences. We also demonstrated that $\mathscr{PP}$ preferences can be elegantly handled in a logic programming framework based on answer set semantics.

The implementation of the language $\mathscr{PP}$ in the **jsmodels** system is almost complete, and this will offer us the opportunity to validate our ideas on large test cases and to compare with related work such as that in Eiter *et al.* (2003). We would also like to develop a direct implementation of the language which can guarantee completeness. In other words, we would like to develop a system that can return *all* possible preferred trajectories.

We also intend to explore the possibility of introducing temporal operators at the level of general preferences. These seem to allow for very compact representation of various types of preferences; for example, a shortest plan preference can be encoded simply as:

$$\textbf{always}((occ(stop) \vee occ(noop)) \lhd (occ(a_1) \vee \ldots \vee occ(a_k)))$$

if $a_1, \ldots, a_k$ are the possible actions. We also intend to natively include in the language preferences like *maxim* used in Section 5.1; these preferences are already expressible in the existing $\mathscr{PP}$ language, but at the expense of large and complex preference formulae. Furthermore, we would like to develop a system that can assist users in defining the preferences given the planning problem.

Acknowledgments

The authors wish to thank the anonymous referees for their valuable comments. The authors were supported by the NSF grants CNS-0220590, CNS-0454066, and HRD-0420407. The authors also wish to thank Hung Le for his implementation of **jsmodels**.

References

Apt, K., Blair, H. and Walker, A. 1988. Towards a theory of declarative knowledge. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 89–148.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Bacchus, F. and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence 116,* 1,2, 123–191.

Bistarelli, S., Codognet, P., Georget, Y. and Rossi, F. 2000. Labeling and Partial Local Consistency for Soft Constraint Programming. *Practical Aspects of Declarative Languages*. Springer Verlag, 230–248.

Bistarelli, S., Montanari, U. and Rossi, F. 1997. Semiring Based Constraint Solving and Optimization. *Journal of the ACM 44,* 2, 201–236.

Blum, A. and Furst, M. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence 90*, 281–300.

Borning, A., Maher, M., Martindale, A. and Wilson, M. 1989. Constraint Hierarchies and Logic Programming. *Proceedings of the International Conference on Logic Programming*. MIT Press, 149–164.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H. and Poole, D. 2004. Preference-based Constrained Optimization with CP-Nets. *Computational Intelligence 20,* 2, 137–157.

Boutilier, C., Brafman, R., Hoos, H. and Poole, D. 1999. Reasoning with Conditional Ceteris Paribus Preference Statements. 15th *Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 71–80.

Brewka, G. 2004a. A Rank Based Description Language for Qualitative Preferences. In *Proceedings of ECAI*, IOS Press, 303–307.

Brewka, G. 2004b. Complex Preferences for Answer Set Optimization. *Proceedings of KR*, AAAI Press, 213–223.

Brewka, G. and Eiter, T. 1999. Preferred answer sets for extended logic programs. *Artificial Intelligence 109*, 297–356.

Brewka, G., Niemelä, I. and Syrjänen, T. 2002. Implementing Ordered Disjunction using Answer Set Solvers for Normal Programs. *Logics in Artificial Intelligence*. Springer Verlag, 444–455.

Brewka, G., Niemelä, I. and Truszczyński, M. 2003. Answer Set Optimization. *International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 867–872.

Buccafurri, F., Leone, N. and Rullo, P. 2000. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering 12,* 5, 845–860.

Cimatti, A. and Roveri, M. 2000. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research 13*, 305–338.

Cui, B. and Swift, T. 2002. Preference Logic Grammars: Fixed Point Semantics and Application to Data Standardization. *Artificial Intelligence 138,* 1–2, 117–147.

Dal Lago, U. and Pistore, M. and Traverso, P. 2002. Planning with a Language for Extended Goals. In *Proceedings of AAAI/IAAI*. AAAI Press, 447–454.

Delgrande, J., Schaub, T. and Tompits, H. 2003. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming 3,* 2 (Mar.), 129–187.

Delgrande, J., Schaub, T. and Tompits, H. 2004. Domain-specific Preferences for Causal Reasoning and Planning. *Principles of Knowledge Representation and Reasoning*. AAAI Press, 673–682.

Dimopoulos, Y., Nebel, B. and Koehler, J. 1997. Encoding planning problems in non-monotonic logic programs. *Proceedings of European Conference on Planning*. Springer Verlag, 169–181.

Domshlak, C. and Brafman, R. 2002. CP-Nets: Reasoning and Consistency Testing. In 8th *International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 121–132.

Eiter, T., Faber, W., Leone, N., Pfeifer, G. and Polleres, A. 2002. Answer Set Planning under Action Cost. *Journal of Artificial Intelligence Research*, *19*, 25–71.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Fargier, H. and Lang, J. 1993. Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach. In *European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty*. Springer Verlag, 97–104.

Gelfond, M. and Lifschitz, V. 1998. Action languages. *Electron. Trans. Artif. Intell. 2*: 193-210.

Gelfond, M., Przymusinska, H. and Przymusinski, T. 1990. On the relationship between CWA, Minimal Model, and Minimal Herbrand Model semantics. *International Journal of Intelligent Systems 5,* 5, 549–565.

Haddawy, P. and Hanks, S. 1993. Utility Model for Goal-Directed Decision Theoretic Planners. Tech. Rep., University of Washington.

Junker, U. 2001. Preference Programming for Configuration. *Proceedings of the IJCAI Workshop on Configuration*. `www.soberit.hut.fi/pdmg/IJCAI2001ConfWS/S`.

Kautz, H. and Walser, J. 1999. State-space Planning by Integer Optimization. In *Proceedings of AAAI*. AAAI Press, 526–533.

Larrosa, J. 2002. On Arc and Node Consistency in Weighted CSP. *Proceedings of AAAI*. AAAI Press, 48–53.

Larrosa, J. and Schiex, T. 2003. In the Quest of the Best Form of Local Consistency for Weighted CSP. *Proceedings of IJCAI*. Morgan Kaufmann, 239–244.

Le, H. V. and Pontelli, E. 2003. A Java Based Solver for Answer Set Programming. `www.cs.nmsu.edu/lldap/jsmodels`.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2005. The DLV System for Knowledge Representation and Reasoning. *ACM Transaction on Computational Logic*. To appear.

Lierler, Y. and Maratea, M. 2004. Cmodels-2: SAT-based Answer Set Solver Enhanced to Non-tight Programs. *Proceedings of the 7th International Conference on Logic Programming and Non-Monotonic Reasoning Conference (LPNMR'04)*. Springer Verlag, 346–350.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence 138,* 1–2, 39–54.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. *Proceedings of the Eleventh International Conf. on Logic Programming*. MIT Press, 23–38.

Lin, F. 1998. On Measuring Plan Quality (A Preliminary Report). *Proceedings of the Sixth International Conferences on Principles of Knowledge Representation and Reasoning (KR'98)*. 224–233.

Lin, F. and Zhao, Y. 2002. ASSAT: Computing Answer Sets of A Logic Program By SAT Solvers. *Proceedings of AAAI*. AAAI Press, 112–117.

Long, D., Fox, M., Smith, D., McDermott, D. Bacchus, F., and Geffner, H. 2002. International Planning Competition. `http://planning.cis.strath.ac.uk/competition/`

Moulin, H. 1988. *Axioms for Cooperative Decision Making*. Cambridge University Press.

Myers, K. 1996. Strategic Advice for Hierarchical Planners. *Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 112–123.

Myers, K. and Lee, T. 1999. Generating Qualitatively Different Plans through Metatheoretic Biases. *Proceedings of AAAI*. AAAI Press, 570–576.

Niemelä, I. 1999. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3,4, 241–273.

Przymusinski, T. 1988. On the declarative semantics of deductive databases and logic programs. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 193–216.

Putterman, M. 1994. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press

Reiter, R. 2001. *KNOWLEDGE IN ACTION: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Schaub, T. and Wang, K. 2001. A Comparative Study of Logic Programs with Preferences. *IJCAI*. Morgan Kaufman, 597–602.

Schiex, T. and Cooper, M. 2002. Constraints and Preferences: The Interplay of Preferences and Algorithms. *Proceedings of the AAAI Workshop on Preferences in AI and CP*.

Schiex, T., Fargier, H. and Verfaillie, G. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems. *Proceedings of IJCAI*. Morgan Kaufmann, 631–637.

Simons, P., Niemelä, I. and Soininen, T. 2002. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence 138,* 1–2, 181–234.

Son, T., Baral, C., Nam, T. and McIlraith, S. 2005. Domain-Dependent Knowledge in Answer Set Planning. *ACM Transaction on Computational Logic*. To Appear.

Son, T. and Pontelli, E. 2004a. Reasoning about Actions and Planning with Preferences using Prioritized Default Theory. *Computational Intelligence 20,* 2, 358–404.

Son, T. and Pontelli, E. 2004b. Planning with Preferences using Logic Programming. *Logic programming and Non-monotonic Reasoning*, Springer Verlag, 247–260.

Subrahmanian, V. and Zaniolo, C. 1995. Relating stable models and ai planning domains. *Proceedings of the International Conference on Logic Programming*. MIT Press, 233–247.

https://doi.org/10.1017/S1471068406002717 Published online by Cambridge University Press