Planning with Preferences using Logic Programming[⋆]

Tran Cao Son   Enrico Pontelli

Department of Computer Science
New Mexico State University
{tson|epontell}@cs.nmsu.edu

Abstract. We present a declarative language, $\mathcal{PP}$, for the 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 them. 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.

1 Introduction

Planning—in its classical sense—is the problem of finding a sequence of actions that achieves a predefined goal. 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 (e.g., see [18]). 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 the 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 [24].

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 features. In these frameworks, it may be simple to find a solution; 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 may be considered acceptable. These issues can be illustrated with the following example:

Example 1. It is 7 am and Bob*, a Ph.D. student, is at home. He needs to be at school at 8 am. He can take a bus, a train, or a taxi to go to school, which will take him 55, 45, or 15 minutes respectively. Taking the bus or the train will require* Bob *to walk to the nearby station, which may take 20 minutes. However, a taxi can arrive in only 5 minutes. When in need of a taxi,* Bob *can call either the* MakeIt50 *or the* PayByMeter *taxi company.* MakeIt50 *will charge a flat rate of $50 for any trip, while* PayByMeter *has a fee schedule of $20 for the trip to school. If he takes the bus or the train, then* Bob *will spend only $2.* Bob*, being a student, prefers to pay less whenever possible. It is easy to see that there are*

⋆ The research has been partially supported by NSF grants EIA0220590, EIA0130887, CCR9875279, CCR9820852, and CCR9900320.

only two feasible *plans for* Bob *to arrive at school on time for his exam: calling one of the two taxi companies. However, a* PayByMeter *taxi would be preferable, as* Bob *wants to save money.* In this case, both plans are feasible but *Bob*'s preference is the deciding factor to select which plan he will follow.

The example shows that users' preferences play a decisive role in the choice of a plan. Thus, we need to be able to evaluate plan components at a finer granularity than simply as consistent or violated. In [20], it is argued that users' preferences are likely more important in selecting a plan for execution, when a planning problem has too many solutions. It is worth observing that, with a few exceptions like the system SIPE-2 with metatheoretic biases [20], 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 [3]. E.g., if Bob's *goal* is to spend at most $2 to go to school, then he does not have any feasible plans to arrive at school on time.

In this paper, we will investigate the problem of integrating users' preferences into a planner. We will develop a high-level language for the specification of user preferences, and then provide a logic programming implementation of the language, based on answer set programming. As demonstrated in this work, normal logic programs with answer set semantics [13] provide a natural and elegant framework to effectively handle planning with preferences. We divide the preferences that a user might have in different categories:

- *Preference 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, even though both satisfy his/her goal;
- *Preference about an action:* the user prefers to perform the action a, whenever it is feasible and it allows the goal to be achieved;
- *Preference 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 about the trajectory, 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). We will also demonstrate how the language for expressing preferences can be realized using Answer Set Programming (ASP).

Related Work: This work is a continuation of our previous work [25], in which we rely on prioritized default theories to express limited classes of preferences between trajectories. This work is also influenced by other works on exploiting *domain-specific knowledge* in planning (e.g., [2, 24]), in which domain-specific knowledge is expressed as a constraint on the trajectories achieving the goal, and hence, is a *hard constraint*.

Numerous approaches have been proposed to integrate preferences in the planning process. Eiter et al. introduced a framework for planning with action costs using logic programming [9]. Each action is assigned an integer cost, and plans with the minimal cost are considered optimal. Costs can be either static or relative to the time step in which the action is executed. [9] also presents 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 [9], but it is more high-level and flexible than the action costs approach. The approach in [9] also does not allow the use of fully general dynamic preferences. Other systems have adopted fixed types of preferences, e.g., shortest plans [6, 4].

Our proposal has similarities with the approach based on metatheories of the planning domain [19, 20], 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.

The problem of maintaining and managing preferences has been investigated in the framework of constraint programming (e.g., [3, 11]). Constraint solving has also been proposed for the management of planning in presence of action costs [15].

Considerable effort has been invested in introducing preferences in logic programming. In [7] 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 [5, 8, 23].

Our language allows the representation of several types of preferences similar to those developed in [14] for decision-theoretic planners. The main 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 [22]; in MDPs, optimal plans are functions from states to actions, thus preventing the user from selecting preferred trajectories without changing the MDP specification.

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)* [17]. 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 will make use of the ontologies of the action description language $\mathcal{B}$ [12]. In $\mathcal{B}$, an action theory is defined over two disjoint sets—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 of actions; I is a set of propositions representing the initial state of the world. For example, the action of calling a taxi has the effect of the taxi arriving, and it is represented in $\mathcal{B}$ as: $call\_taxi$ **causes** $taxi\_arrived$. Realistically, should one need to execute this action one has to have enough money. This is expressed in $\mathcal{B}$ by the proposition: $call\_taxi$ **executable_if** has_enough_money. In this paper, we will assume that I is complete, i.e., for every fluent $f \in$ **F**, I contains either f or $\neg f$.

The semantics of an action theory is given by the notion of a *state*—a consistent set of fluent literals (i.e., fluents and negated fluents) that satisfies the relationship between fluents—and a *transition function* Φ that specifies the result of the execution of an action a in a state s, denoted by $\Phi(a, s)$. A *trajectory* of an action theory (D, I) is a sequence $s_0 a_1 s_1 \ldots a_n s_n$ 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 state s satisfies a fluent literal f, denoted by $s \models f$, if $f \in s$. Since our main concern in this paper is not the language for representing actions and their effects, we omit here the detailed definition of $\mathcal{B}$ [12].

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 based on fluent literals) representing the goal. A possible solution to $\langle D, I, G\rangle$ is a trajectory $s_0 a_1 s_1 \ldots a_m s_m$, where $s_0 \models I$ and $s_m \models G$. In this case, we say that the trajectory achieves G.

Answer set planning [17] solves a planning problem $\langle D, I, G\rangle$ by translating it into a logic program $\Pi(D, I, G)$ consisting of *domain-dependent* rules that describe D, I, and G and *domain-independent* rules that generate action occurrences and represent the transitions between states. Besides the planning problem, $\Pi(D, I, G)$ requires an additional parameter: 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 t. Details about the program $\Pi(D, I, G)$ can be found in [24]. 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 [24]. Answer sets of the program $\Pi(D, I, G)$ can be computed using answer set solvers such as **smodels** [21], **dlv** [10], **cmodels** [1], or **jsmodels** [16].

3 A Language for Planning Preferences Specification

In this section, we introduce the language $\mathcal{PP}$ for planning preference specification. Let $\langle D, I, G\rangle$ be a planning problem, with actions **A** and fluents **F**; let $\mathcal{F}_F$ be the set of all fluent formulae. $\mathcal{PP}$ is defined as special formulae over **A** and **F**. We subdivide preferences in different classes: *basic desires*, *atomic preferences*, and *general preferences*.

3.1 Basic Desires

A basic desire is a formula expressing a preference about a trajectory. For example, *Bob*'s basic desire is to save money; this implies that he prefers to use the train or the bus to go to school, which, in turn, means that a preferred trajectory for *Bob* should contain the action $take\_bus$ or $take\_train$. This preference could also be expressed by a formula that forbids the fluent $taxi\_arrived$ to become true in every state of the trajectory. These two alternatives of preference representation are not always equivalent. The first one represents the desire of leaving a state by 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 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 are formally defined next.

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

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.*

To express that *Bob* would like to take the train or the bus to school, we can write:

$$\textbf{eventually}(occ(take\_bus) \vee occ(take\_train)).$$

If *Bob* does not desire to call a taxi, we can write: **always**$(\neg occ(call\_taxi))$. We could also write: **always**$(\neg taxi\_arrived)$. 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 specify when a trajectory satisfies a basic desire formula. In a later section, 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*

- $\varphi =$ ***goal***(ψ) *and $s_n \models \psi$*
- *$\varphi = \psi \in \mathcal{F}_F$ and $s_0 \models \psi$*
- *$\varphi = occ(a)$, $a_1 = a$, and $n \geq 1$*
- *$\varphi = \psi_1 \wedge \psi_2$, $\alpha \models \psi_1$ 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 $n \geq 1$*
- $\varphi =$ **always**(ψ) *and $\forall(0 \leq i \leq n)$ we have that $\alpha[i] \models \psi$*
- $\varphi =$ **eventually**(ψ) *and $\exists(0 \leq i \leq n)$ such that $\alpha[i] \models \psi$*
- $\varphi =$ **until**(ψ_1, ψ_2) *and $\exists(0 \leq i \leq n)$ such that $\forall(0 \leq j < i)$ we have that $\alpha[j] \models \psi_1$ and $\alpha[i] \models \psi_2$.*

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 (denoted as $\alpha \approx_\varphi \beta$) if one of the two following cases occur: (i) *$\alpha \models \varphi$ and $\beta \models \varphi$, or* (ii) *$\alpha \not\models \varphi$ 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;

Definition 5 (Weak Single Desire Preference). *Let φ be a 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 $\alpha \approx_\varphi \beta$.*

Proposition 1. *The relation $\preceq_\varphi$ defines a partial order over the trajectories.*

These 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 sugars to simplify their use:

- (Strong Desire) given the desire formulae φ_1, φ_2, $\varphi_1 < \varphi_2$ denotes $\varphi_1 \wedge \neg\varphi_2$.
- (Weak Desire) given the 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 $executable(a_1) \wedge executable(a_2) \Rightarrow occ(a_1) < occ(a_2)$. This can be extended to include disjunction (or group) of actions on each side of the formula.

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 preference, a most preferred trajectory might require extra actions that would have been otherwise considered unnecessary as shown below.

Example 2. Let us enrich the theory from Example 1 with an action called *buy_coffee*, which allows Bob to have coffee. He can do it only at the station. To say that *Bob* prefers to have coffee before he takes the exam, we write: **goal**$($*have_coffee*$)$. Any plan satisfying this preference requires *Bob* to stop at the station before taking the exam. E.g., while *calls a taxi* and then *takes the taxi to school* is a valid trajectory for *Bob* to achieve his goal, this is not a most preferred trajectory; instead, *Bob* has to take the taxi to the station, buy the coffee, and then go to school. Besides the action of buy_coffee that is needed for *Bob* to get the coffee, the most preferred trajectory requires the action of *driving to the station*, which is not necessary if *Bob* does not have the preference of having the coffee.

3.2 Atomic Preferences and Chains

Basic desires allow 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 desire is to find a trajectory that satisfies all formulae. In many cases, this proves to be too strong and results in situations where no preferred trajectory can be found. For example, *time* and *cost* are often two criteria that a person might have when making a travel plan. This two criteria are often in conflict, i.e., transportation method that takes little time often costs more. It is very unlikely that he/she can get a plan that can satisfy both criteria. Consider Example 1, it is obvious that *Bob* cannot have a plan that costs him only two dollars and allows him to get to destination quickly. To address this problem, we allow a new type of formulae, *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 $\varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_n$ ($n \geq 1$) 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. The definitions of $\approx$ and $\prec$ are 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 $\forall i.[1 \leq i \leq n \Rightarrow \alpha \approx_{\varphi_i} \beta]$.*

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

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

We can show that this version of $\preceq$ is a partial order (with $\approx$ as underlying equivalence).

Proposition 2. *Let Ψ be an atomic preference; then $\preceq_\Psi$ is a partial order.*

A trajectory α is most preferred if there is no other trajectory that is preferred to α.

Example 3. Let us continue with the theory in Example 2. To simplify the representation, we will assume that each action is associated with a degree of safety. We will also write bus, $train$, $taxi_1$, $taxi_2$, and $walk$ to say that Bob takes the bus, train, taxi with *PayByMeter* or *MakeIt50*, or walk, respectively. The following is a desire expressing that *Bob* prefers to get the fastest possible way to go to school:

$$time = \textbf{always}(taxi_1 \vee taxi_2 <^e bus \vee train \vee walk)$$

On the other hand, when he is not in a hurry, *Bob* prefers to get the cheaper way to go to school: $cost = \textbf{always}(walk \vee bus \vee train <^e taxi_1 \vee taxi_2)$

These two preferences 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 *Bob* when he is in a hurry while the second one is more appropriate for *Bob* when he has time. The trajectory $\alpha = s_0\ walk\ s_1\ bus\ s_2$ is preferred to the trajectory $\beta = s_0\ call\_taxi(PayByMeter)\ s'_1\ taxi_1\ s'_2$ with respect to the preference $cost \lhd time$, i.e., $\alpha \prec_{cost \lhd time} \beta$. (for brevity, we omit the description of the states s_i's.) However, $\beta \prec_{time \lhd cost} \alpha$.

3.3 General Preferences

A general preference is constructed from atomic preferences as follows.

Definition 9 (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;*
- *Given a collection of general preferences $\Psi_1, \Psi_2, \ldots, \Psi_k$, then $\Psi_1 \lhd \Psi_2 \lhd \cdots \lhd \Psi_k$ is a general preference.*

Intuitively, the operators $\&, |, !$ are used to express different ways to combine preferences. Syntactically, they are similar to the operations $\wedge, \vee, \neg$ in the construction of basic desire formulae. Semantically, they differ from the operations $\wedge, \vee, \neg$ in a subtle way. For example, given two fluent formulae ϕ and ψ, it is easy to see that both $\phi \vee \psi$ and $\phi \mid \psi$ are general preferences. Although both express our preference over trajectories, the first formula represents a *single preference* while the second one provides *two different criteria* and we have no preference between them.

Definition 10 (Ordering Between Trajectories w.r.t. General Preferences). *Let Ψ a general preference and α, β two trajectories.*

- *The trajectory α is preferred to β ($\alpha \prec_\Psi \beta$) if:*
  - *Ψ is an atomic preference and $\alpha \prec_\Psi \beta$*
  - *$\Psi = \Psi_1 \& \Psi_2$, $\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$*
  - *$\Psi = \Psi_1 \mid \Psi_2$ and:* (i) *$\alpha \prec_{\Psi_1} \beta$ and $\alpha \approx_{\Psi_2} \beta$; or* (ii) *$\alpha \prec_{\Psi_2} \beta$ and $\alpha \approx_{\Psi_1} \beta$; or* (iii) *$\alpha \prec_{\Psi_1} \beta$ and $\alpha \prec_{\Psi_2} \beta$*

– *$\Psi =! \Psi_1$ and $\beta \prec_{\Psi_1} \alpha$ or $\alpha \approx_{\Psi_1} \beta$*
– *$\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$, and there exists $1 \leq i \leq k$ such that:* (i) *$\forall (1 \leq j < i)$ we have that $\alpha \approx_{\Psi_j} \beta$, and* (ii) *$\alpha \prec_{\Psi_i} \beta$.*

- *The trajectory α is indistinguishable from the trajectory β ($\alpha \approx_\Psi \beta$) if:*
  – *Ψ is an atomic preference and $\alpha \approx_\Psi \beta$.*
  – *$\Psi = \Psi_1 \& \Psi_2$, $\alpha \approx_{\Psi_1} \beta$, $\alpha \approx_{\Psi_2} \beta$.*
  – *$\Psi = \Psi_1 \mid \Psi_2$, $\alpha \approx_{\Psi_1} \beta$, and $\alpha \approx_{\Psi_2} \beta$.*
  – *$\Psi =! \Psi_1$ and $\alpha \approx_{\Psi_1} \beta$.*
  – *$\Psi = \Psi_1 \lhd \cdots \lhd \Psi_k$, and for all $1 \leq i \leq k$ we have that $\alpha \approx_{\Psi_i} \beta$.*

Again, we can prove that $\approx_\Psi$ is an equivalence relation and $\preceq_\Psi$ is a partial ordering.

Proposition 3. *If Ψ is a general preference, then $\approx_\Psi$ is an equivalence relation.*

Proposition 4. *Let Ψ be a general preference. Then $\prec_\Psi$ is a transitive relation and the relation $\preceq$ is a partial order (with $\approx$ as base equivalence).*

A trajectory α is most preferred if there is no trajectory that is preferred to α.

Example 4. Let us continue with the theory of Example 3. Assume that the safest transportation mode is either the *train* or the expensive *MakeIt50* cab. The preference

$$safety = \textbf{always}(train \vee walk \vee taxi_2 <^e bus \vee taxi_1)$$

says that *Bob* prefers to move around using the safest transportation mode. Further, he prefers safety over time and cost, so we write $safety \lhd (time \& cost)$.

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 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 lost of generality, we can assume that we only have one preference formula to deal with. Given a planning problem $\langle D, I, G\rangle$ and a preference formula φ, we are interested in finding a preferred trajectory α achieving G for φ. We will show how this can be done in answer set programming. We achieve that by encoding each basic desire φ as a set of rules Π_φ and developing two sets of rules Π_{sat} and Π_{pref}. Π_{sat} checks whether a basic desire is satisfied by a trajectory. Π_{pref} consist of rules that, when used with the **maximize** construct of **smodels** will 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 begin by defining Π_φ.

4.1 Encoding of Desire Formulae

The encoding of a desire formula is similar to the encoding of a fluent formula in [24]. In our encoding, we will use the predicate $formula$ as a domain predicate. The set $\{formula(l, l) \mid l$ is a fluent literal$\}$ will belong to Π_φ. Each of the atoms in this set declares that each literal is also a formula. Next, each basic desire formula φ will be associated with a unique name n_φ. If φ is a fluent formula then it is encoded by a set of atoms of the form $formula(.,.)$ and is denoted by r_φ. For example, $\varphi = f \wedge g$ will be given a name, $n_{f\wedge g}$, and is encoded by the formula $formula(n_{f\wedge g}, conjunction(f, g))$. For other types of desire formula, Π_φ is defined as follows.

– If $\varphi = goal(\phi)$ then $\Pi_\varphi = r_\phi \cup \{desire(n_\varphi), goal(n_\phi)\}$;
– If φ is a fluent formula then $\Pi_\varphi = r_\varphi \cup \{desire(n_\varphi)\}$;
– 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 $\varphi =$ **next**(ϕ) then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), next(n_\varphi, n_\phi)\}$;
– If $\varphi =$ **until**(φ_1, φ_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 $\varphi =$ **always**(ϕ) then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), always(n_\varphi, n_\phi)\}$;
– If $\varphi =$ **eventually**(ϕ) then $\Pi_\varphi = \Pi_\phi \cup \{desire(n_\varphi), eventually(n_\varphi, n_\phi)\}$.

4.2 Π_{sat} – Rules for Checking of Basic Desire Formula Satisfaction

We now present the set of rules that check 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 $length$. 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 is defined based on the 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. They are defined based on the structure of F. Some of the rules of Π_{sat} are given next.

$$satisfy(F, T) \leftarrow desire(F), goal(F), satisfy(F, length). \quad (1)$$
$$satisfy(F, T) \leftarrow desire(F), happen(F, A), occ(A, T). \quad (2)$$
$$satisfy(F, T) \leftarrow desire(F), formula(F, G), h(G, T). \quad (3)$$
$$satisfy(F, T) \leftarrow desire(F), and(F, F_1, F_2), satisfy(F_1, T), satisfy(F_2, T). \quad (4)$$
$$satisfy(F, T) \leftarrow desire(F), negation(F, F_1), not\ satisfy(F_1, T). \quad (5)$$
$$satisfy(F, T) \leftarrow desire(F), until(F, F_1, F_2), during(F_1, T, T_1), satisfy(F_2, T_1). \quad (6)$$
$$satisfy(F, T) \leftarrow desire(F), always(F, F_1), during(F_1, T, length{+}1). \quad (7)$$
$$satisfy(F, T) \leftarrow desire(F), next(F, F_1), satisfy(F_1, T + 1). \quad (8)$$
$$during(F_1, T, T_1) \leftarrow T < T_1 - 1, desire(F_1), satisfy(F_1, T), during(F_1, T + 1, T_1). \quad (9)$$
$$during(F_1, T, T_1) \leftarrow T = T_1 - 1, desire(F_1), satisfy(F_1, T). \quad (10)$$

In the next theorem, we prove the correctness of Π_{sat}. We need some additional notation. Let M be an answer set of the program $\Pi(D, I, G)$. By α_M we denote the trajectory $s_0 a_0 \ldots a_{n-1} s_n$, where (i) $occ(a_i, i) \in M$ for $i \in \{0, \ldots, n-1\}$ and (ii) $s_i = \{f \mid holds(f, i) \in M\}$ for $i \in \{0, \ldots, n\}$. For a trajectory $\alpha = s_0 a_0 \ldots a_{n-1} s_n$, let $\alpha^{-1} = \{occ(a_i, i) \mid i \in \{0, \ldots, n-1\} \cup \{holds(f, i) \mid f \in s_i, i \in \{0, \ldots, n\}$. We have:

Theorem 1. *Let $\langle D, I, G\rangle$ be a planning problem and φ be a basic desire formula. Let M be an answer set of $\Pi(D, I, G)$. Then, $\alpha_M \models \varphi$ iff $\Pi_\varphi \cup \Pi_{sat} \cup (\alpha_M)^{-1} \models satisfy(n_\varphi, 0)$.*

The theorem allows us to compute a most preferred trajectory in **smodels**. 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

$$\mathbf{maximize}\{satisfy(n_\varphi, 0) = 1, not\ satisfy(n_\varphi, 0) = 0\}. \quad (11)$$

Note that rule (11) means that 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 when no answer set with this property exists, other answer sets are considered. (The current implementation of **smodels** has some restrictions on using the **maximize** construct; our **jsmodels** system can now deal with this construct properly.)

Theorem 2. *Let $\langle D, I, G\rangle$ be a planning problem and φ be a basic desire formula. For every answer set M of $\Pi(D, I, G, \varphi)$, 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 general preferences using the **maximize** function. The intuition is to associate to the different components of 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** function can be used to handle these weights and guide the search of the preferred trajectory. In general, let Ψ be a general preference. We will develop a *weight function*, w_Ψ, which maps each trajectory to a number and 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 $w_\Psi(\alpha) = w_\Psi(\beta)$. A weight function satisfying the two properties (i)-(ii) is called an *admissible* weight function. Obviously, if w_Ψ is admissible, we have the following theorem.

Proposition 5. *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.*

The above proposition implies that we can compute a most preferred trajectory using **smodels** if we can implement an admissible weight function.

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 basic desires.

Definition 11 (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 desire φ is a function defined as*
$w_\varphi(\alpha) = 1$ if $\alpha \models \varphi$ and $w_\varphi(\alpha) = 0$, otherwise.

The next proposition shows that for a basic desire φ, w_φ is admissible.

Proposition 6. *Let φ be a basic desire. Then w_φ is an admissible weight function.*

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

Definition 12 (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:*

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

Again, we can show that the above function is admissible.

Proposition 7. *Let $\psi = \varphi_1 \lhd \varphi_2 \lhd \cdots \lhd \varphi_k$ be an atomic preference. Then w_ψ is an admissible weight function.*

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

Definition 13 (General Preference Weight). *Let Ψ be a general preference formula. The weight of a trajectory α w.r.t. Ψ ($w_\Psi(\alpha)$) is defined as follows:*

- *if Ψ is an atomic preference then the weight is defined as in definition 12.*
- *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.*
- *if $\Psi = \Psi_1 \lhd \Psi_2$[1] 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 8. *If Ψ is a general preference, then w_Ψ is an admissible weight function.*

Propositions 6-8 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. Similar to the encoding of desires, we will assign a new, distinguished name n_ϕ to each preference formula ϕ, which is not a desire, occurring in Ψ and encode the preferences in the same way we encode the desires. To save space, we omit here the details of this step. $\Pi_{pref}(\Psi)$ define 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 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 desire d, $\Pi_{pref}(d)$ contains the rules
   $w(d, 1) \leftarrow satisfy(d)$. $\quad w(d, 0) \leftarrow not\ satisfy(d)$. $\quad max(d, 2) \leftarrow$.
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 next two rules:
   $w(n_\phi, S) \leftarrow w(n_{\varphi_1}, N_1), w(n_{\varphi_2}, N_2), \ldots, w(n_{\varphi_k}, N_k), S = \Sigma_1^k 2^{k-r} N_r$.
   $max(n_\phi, 2^{k+1} + 1) \leftarrow$.
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
  $w(n_\Psi, S) \leftarrow w(n_{\Psi_1}, N_1), w(n_{\Psi_2}, N_2), S = N_1 + N_2$.
  $max(n_\Psi, S) \leftarrow max(n_{\Psi_1}, N_1), max(n_{\Psi_2}, N_2), S = N_1 + N_2$.
- if $\Psi = !\,\Psi_1$ then $\Pi_{pref}(\Psi)$ consists of $\Pi_{pref}(\Psi_1)$ and the rules
  $w(n_\Psi, S) \leftarrow w(n_{\Psi_1}, N), max(n_{\Psi_1}, M), S = M + 1 - N$.
  $max(n_\Psi, S) \leftarrow max(n_{\Psi_1}, M), S = M + 1$.
- if $\Psi = \Psi_1 \lhd \Psi_2$ then $\Pi_{pref}(\Psi)$ consists of $\Pi_{pref}(\Psi_1) \cup \Pi_{pref}(\Psi_2)$ and rules
  $w(n_\Psi, S) \leftarrow w(n_{\Psi_1}, N_1), max(n_{\Psi_2}, M_2), w(n_{\Psi_2}, N_2), S = M_2 * N_1 + N_2$.
  $max(n_\Psi, S) \leftarrow max(n_{\Psi_1}, N_1), max(n_{\Psi_2}, N_2), S = N_2 * N_1 + N_2 + 1$.

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

Theorem 3. *Let Ψ be a general preference. For every answer set M of $\Pi(D, I, G)$ we have that $\Pi_{pref}(\Psi) \cup \Pi_{sat} \cup (\alpha_M)^{-1} \models w(n_\Psi, w)$ iff $w_\Psi(\alpha_M) = w$.*

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**.

[1] Without loss of generality, we describe the encoding only for chains of length 2.

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 [9]. Let $\langle D, I, G\rangle$ be a planning problem. In keeping with the notation used in the previous section, we use φ to denote G (i.e., $\varphi = G$).

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 $\sigma^i(\varphi)$ ($0 \leq i \leq n$) as follows:

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

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

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

Proposition 9. *Let α be a most preferred trajectory w.r.t. $short(n,\varphi)$. Then α is a shortest length trajectory satisfying the goal φ.*

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 fictions 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 is achieved. Again, we appeal to the users for the formal representation of these actions. Furthermore, we add the condition $\neg ended$ to the executability condition of any actions in (D, I) and to the initial state I. Then we can encode the condition of shortest length trajectory, denoted by $short$, as

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

where $a_1,\ldots,a_k$ are the actions in the original action theory.

Proposition 10. *Let α be a most preferred trajectory w.r.t. $short$. Then α is a shortest length trajectory satisfying the goal φ.*

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 only on equal length trajectories, 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. Further, we introduce the fluent $sCost$ to denote the cost of the trajectory. Initially, we set the value of $sCost$ to 0 and the execution of action a will increase the value of $sCost$ by $c(a)$. The preference **goal**$(sCost(m)) \lhd$ **goal**$(sCost(m+1))\ldots \lhd$ **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 Conclusion

In this paper we presented a novel declarative language, called $\mathcal{PP}$, for the specification of preferences in the context of planning problems. The language nicely integrates with traditional action description languages (e.g., $\mathcal{B}$) and it allows 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 encoding—as in the approaches based on utility functions). $\mathcal{PP}$

allows the expression of complex preferences, including multi-dimensional preferences. We also demonstrated that $\mathcal{PP}$ preferences can be effectively and easily handled in a logic programming framework based on answer set semantics.

The work is still in its preliminary stages. The implementation of the required cost functions 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 [9]. 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.

References

1. Y. Babovich. “CMODELS”, `www.cs.utexas.edu/users/tag/cmodels.html`.
2. F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1,2):123–191, 2000.
3. S. Bistarelli et al. Labeling and Partial Local Consistency for Soft Constraint Programming. In *Practical Aspects of Declarative Languages*, Springer Verlag, 2000.
4. A.L. Blum and M.L. Furst. Fast Planning through Planning Graph Analysis. *AIJ*, 90, 1997.
5. G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *AIJ*, 109, 1999.
6. A. Cimatti and M. Roveri. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.
7. B. Cui and T. Swift. Preference Logic Grammars: Fixed Point Semantics and Application to Data Standardization. *Artificial Intelligence*, 138(1–2):117–147, 2002.
8. J. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, March 2003.
9. T. Eiter et al. Answer Set Planning under Action Cost. In *JELIA*. Springer Verlag, 2002.
10. T. Eiter et al. The KR System `dlv`: Progress Report, Comparisons, and Benchmarks. In *Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 406–417, 1998.
11. F. Fages, J. Fowler, and T. Sola. Handling Preferences in Constraint Logic Programming with Relational Optimization. In *PLILP* Springer Verlag, 1994.
12. M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.
13. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. ICLP’88.
14. P. Haddawy and S. Hanks. Utility Model for Goal-Directed Decision Theoretic Planners. Technical report, University of Washington, 1993.
15. H. Kautz and J.P. Walser. State-space Planning by Integer Optimization. In *AAAI*, 1999.
16. H. Le, E. Pontelli, T.C. Son. A Java Solver for Answer Set Programming, NMSU, 2003.
17. V. Lifschitz. Answer set planning. In *ICLP*, pages 23–37, 1999.
18. D. Long et al. International Planning Competition.
19. K.L. Myers. Strategic Advice for Hierarchical Planners. In *KR*’96.
20. K.L. Myers and T.J. Lee. Generating Qualitatively Different Plans through Metatheoretic Biases. In *AAAI*, 1999.
21. I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *LPNMR*, Springer, pages 420–429, 1997.
22. M.L. Putterman. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. John Willey & Sons, Inc., New York, NY, 1994.
23. T. Schaub et al. A Comparative Study of Logic Programs with Preferences. *IJCAI*, 2001.
24. T.C. Son, C. Baral, and S. McIlraith. Domain dependent knowledge in planning - an answer set planning approach. In *LPNMR*, Springer, pages 226–239, 2001.
25. T.C. Son and E. Pontelli. Reasoning about actions in prioritized default theory. In *JELIA*, 2002.