

Reasoning About Actions in Prioritized Default Theory

Tran Cao Son and Enrico Pontelli

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

Abstract. This paper shows how action theory in the language \mathcal{B} can be naturally encoded using prioritized default theory. We also show how prioritized default theory can be extended to express preferences between *rules* and *formulae*. This extension provides a natural framework to introduce *preferences over trajectories* in \mathcal{B} . We illustrate how these preferences can be expressed and how they can be represented within extended prioritized default theory. We also discuss how this framework can be implemented in terms of answer set programming.

1 Introduction

Research in reasoning about action and change (RAC) has been concentrated on developing formalisms for representing and reasoning about actions and their effects. In general, an action theory is a set of propositions written in a specialized language such as situation calculus [12], event calculus [7], STRIPS [2], action description language [15, 4] etc. that was developed for representing and reasoning about actions and their effects. The semantics of an action theory is then defined by an entailment relation that determines what will be true after an action sequence is executed from a given state.

For several years, the frame problem [12], the ramification problem [6], and the qualification problem [11] have been at the center of RAC's research. In short, the frame problem is the problem of describing, in a concise way, the non-effects of actions, i.e., to express what *does not change* after an action is executed. The ramification problem is concerned with the representation of static domain constraints or indirect effects of actions. The qualification problem, on the other hand, is concerned with actions that may not be executable in a certain situation. To date, solutions to these problems have been discussed in several RAC's approaches.

In this paper we show that prioritized default theory [5] provides a natural framework for representing and reasoning about actions and their effects. We show that by viewing dynamic and static causal laws as rules and the inertial law as defaults, action theories—in this work we concentrate on the language \mathcal{B} [4]—can be translated into semantically equivalent prioritized default theories. The novelty of this work is a methodology of translating action theories into prioritized default theory. Furthermore, we propose an extension of prioritized default theory in which preferences between rules and formulae can be enforced in the process of proving consequences. The advantage of this new formalism is that it provides a convenient way to incorporate different forms of *preferences* in the process of representing and reasoning about trajectories (or plans in deterministic action theories). Conventional logic-based approaches to reasoning about actions provide the ability to derive trajectories leading to states satisfying a

predefined goal. Nevertheless, in many situations, it is desirable to find one, among several possible trajectories, that satisfies certain constraints. Thus, we would like to allow the action language to include a description of users' preferences in how to accomplish the goal. In this work we explore alternative forms of preferences at the level of the action language: (1) preferences between actions; (2) preferences between final states; (3) general preferences between trajectories. These preferences could be viewed as *soft constraints* on a trajectory or a plan, that may or may not be satisfied depending on the particular situation. We illustrate how these different forms of preferences can be represented and handled in the context of prioritized default theories. As a consequences, prioritized default theory and its implementation in logic programming automatically provide an effective methodology to derive preferred trajectories.

2 Background

2.1 The Action Language \mathcal{B}

In this section, we present the basic terminology associated to the language \mathcal{B} [4]. The alphabet of \mathcal{B} consists of two nonempty, disjoint sets, \mathbf{F} and \mathbf{A} , of *fluent names* and *action names*, respectively. A *fluent literal* is a fluent name possibly preceded by \neg . A *domain description* (or *domain*) is a set of propositions of the forms

$$a \quad \mathbf{causes} \quad f \quad \mathbf{if} \quad p_1, \dots, p_n \quad (1)$$

$$f \quad \mathbf{if} \quad p_1, \dots, p_m \quad (2)$$

$$a \quad \mathbf{executable_if} \quad p_1, \dots, p_n \quad (3)$$

where a is an action name and f and p_i 's are fluent literals. They are called *dynamic*, *static laws*, and *executability condition* respectively. The dynamic law (1) represents the effect of action a on fluent f while the causal law (2) states that the fluent f will be true in any state in which p_1, \dots, p_m are true. The proposition (3) represents the conditions under which the action a can be executed. Axioms in \mathcal{B} are propositions of the form

$$\mathbf{initially} \quad f \quad (4)$$

where f is a fluent literal. This axiom states that the fluent literal f is true in the initial state. Finally, a query in \mathcal{B} is of the form

$$\varphi \quad \mathbf{after} \quad \alpha \quad (5)$$

where φ is a fluent formula and α is a sequence of actions. Intuitively, the query asks whether the fluent formula φ is true in all the states resulting from the execution of the action sequence α from the current state. An *action theory* is a pair (D, Γ) , where D is a collection of propositions of the type (1)–(3), called a *domain description*, while Γ is a collection of propositions of type (4), called a *initial state*. The next example illustrates the use of the language \mathcal{B} in describing a simple dynamic domain.

Example 1. Consider the problem of describing the functioning of a briefcase [20]; the briefcase is provided with two clasps, and it will not open unless both clasps are

unfastened. The domain description makes use of the fluents

open *briefcase is open*
fastened(X) *clasp X is fastened*

We provide a single domain action to change the status of the clasp:

unfastened(X) **causes** \neg fastened(X)

Finally, a static causal law is used to propagate the effect of unfastening both clasps:

open **if** \neg fastened(*clasp*₁), \neg fastened(*clasp*₂)

A domain description given in \mathcal{B} defines a transition function (Φ) from actions and states to a set of states. Intuitively, given an action a and a state s , the function Φ defines the set of states $\Phi(a, s)$ that may be reached after executing the action a in state s . If $\Phi(a, s)$ is an empty set it means that a is not executable in s .

Let D be a domain description in \mathcal{B} . An *interpretation* I of the fluents in D is a maximal consistent set of fluent literals from \mathbf{F} . A fluent f is said to be true (resp. false) in I iff $f \in I$ (resp. $\neg f \in I$). The truth value of a fluent formula in I is defined recursively over the propositional connectives in the usual way. For example, $f \wedge q$ is true in I iff f is true in I and q is true in I . We say that a formula φ holds in I (or I satisfies φ), denoted by $I \models \varphi$, if φ is true in I .

Let u be a consistent set of fluent literals and K a set of static causal laws. We say that u is closed under K if for every static causal laws “**if** ($\{p_1, \dots, p_n\}, f$)” in K , if $\{p_1, \dots, p_n\} \subseteq u$ then so does f . By $Cl_K(u)$ we denote the least consistent set of literals from D that contains u and is also closed under K . A *state* of D is an interpretation of the fluents in \mathbf{F} that is closed under the set of static causal laws belonging to D . An action a is *executable* in a state s if there exists a proposition

a executable if f_1, \dots, f_n

in D such that $s \models f_1 \wedge \dots \wedge f_n$. If “ **a executable if true**” belongs to D , then a is executable in every state of D . The *immediate effect of an action* a in state s is the set

$$E(a, s) = \{f \mid D \text{ contains a dynamical law “} a \text{ causes } f \text{ if } f_1, \dots, f_n\text{” and } s \models f_i \text{ for } i = 1, \dots, n\}.$$

For a domain description D , $\Phi(a, s)$, the set of states that may be reached by executing a in s , is defined as follows: if a is executable in s , then

$$\Phi(a, s) = \{s' \mid s' \text{ is a state and } s' = Cl_{D_C}(E(a, s) \cup (s \cap s'))\}$$

where D_C is the set of static causal laws in D ; if a is not executable in s , then $\Phi(a, s) = \emptyset$. Every domain description D in \mathcal{B} has a unique transition function Φ , and we say Φ is the transition function of D .

For a domain D with transition Φ , a sequence $s_0 a_1 s_1 \dots a_n s_n$ where s_i 's are states and a_i 's are actions is called a *trajectory* in D if $s_{i+1} \in \Phi(s_i, a_{i+1})$ for $i \in \{0, \dots, n-1\}$. A trajectory $s_0 a_1 s_1 \dots a_n s_n$ is a trajectory of a fluent formula Δ if $s_n \models \Delta$.

A domain description D is *consistent* iff for every action a and state s , if a is executable in s , then $\Phi(a, s) \neq \emptyset$. An action theory (D, Γ) is consistent if D is consistent and $s_0 = \{f \mid \text{“initially } f\text{”} \in \Gamma\}$ is a state of D . An action theory (D, Γ) is *complete* if, for each fluent f , we have that either “**initially** f ” or “**initially** $\neg f$ ” belongs to Γ .

s_0 will be called the initial state of (D, Γ) . Finally, for an action theory (D, Γ) whose initial state s_0 and an action sequence $\alpha = a_1, \dots, a_n$, we say that the query φ **after** α is entailed by (D, Γ) , denoted by $(D, \Gamma) \models \varphi$ **after** α if for every possible trajectory $s_0 a_1 s_1 \dots a_n s_n$, φ holds in s_n .

In what follows, we will consider only consistent and complete action theories. We will also omit the description of the set of action names and the set of fluent names.

Example 2. Let D_b be the domain description in Example 1 and Γ be the initial state contains the three propositions

initially \neg open,
initially \neg fastened(*clasp*₁), and
initially \neg fastened(*clasp*₂).

The initial state s_0 in this action theory is

$\{\neg$ open, \neg fastened(*clasp*₁), \neg fastened(*clasp*₂) $\}$.

The action *unfastened*(*clasp*₁) is executable in s_0 . We can easily check that

if $s \in \Phi(\text{unfastened}(\textit{clasp}_1), s_0)$ then $s \models \neg$ fastened(*clasp*₁).

This implies that $(D_b, \Gamma) \models \neg$ fastened(*clasp*₁) **after** *unfastened*(*clasp*₁). If we consider also the action *unfastened*(*clasp*₂) then we can obtain

if $s_1 \in \Phi(\text{unfastened}(\textit{clasp}_2), s_2)$ and $s_2 \in \Phi(\text{unfastened}(\textit{clasp}_1), s_0)$
then $s_1 \models$ open.

Hence, $(D_b, \Gamma) \models$ open **after** *unfastened*(*clasp*₁), *unfastened*(*clasp*₂).

2.2 Prioritized Default Theory

Prioritized default theory has been discussed in [5]. In this paper we decided to rely on prioritized default theory because of two major reasons. First of all, its syntax is simple and intuitive. Furthermore, the semantics of prioritized default theory is defined in terms of logic programs and answer set semantics [3]. Not only this avoids the creation of an ad-hoc semantics, but this also allows us to reuse existing inference systems developed for answer set semantics (e.g., **smodels** and **dlv**) to compute the entailment relation of prioritized default theory. In this paper we begin with the theory proposed by Gelfond and Son in [5]. We then extend it to allow new type of preferences such as those between rules or formulas to be encoded.

A prioritized default theory comprises of facts, defaults, rules, and preferences between defaults. Rules and defaults are used to derive new conclusions. Nevertheless, the use of rules and defaults is different. A rule is used to derive a conclusion whenever all its premises are satisfied. On the other hand, a default can be used to derive a conclusion as long as such conclusion does not introduce inconsistencies into the theory—even if

all its premises are satisfied. Formally, a default theory over a multi-sorted logic language \mathcal{L} (or a *domain*) is a set of literals of the form

$$rule(r, l_0, [l_1, \dots, l_m]); \quad (6)$$

$$default(d, l_0, [l_1, \dots, l_m]); \quad (7)$$

$$prefer(d_1, d_2); \quad (8)$$

where r is a rule name, d, d_1, d_2 are default names, l_0, \dots, l_n are literals of the language \mathcal{L} , and $[]$ is the list operator. For a rule r , let $body(r)$ denote the list $[l_1, \dots, l_n]$ and let $head(r)$ denote the literal l_0 . Similar notation will be used for defaults. We assume that default names and rule names are two disjoint sets. The semantics of a default theory T is defined by the answer set semantics of a logic program, consisting of T and the following set of independent axioms:

- **Rules for Inference:**

$$holds(L) \leftarrow rule(R, L, Body), hold(Body). \quad (9)$$

$$holds(L) \leftarrow default(D, L, Body), hold(Body), \quad (10)$$

$$not\ defeated(D), not\ holds(\neg L).$$

$$hold([]) \leftarrow \quad (11)$$

$$hold([H|T]) \leftarrow holds(H), hold(T). \quad (12)$$

- **Rules for Defeating Defaults:**

$$defeated(D) \leftarrow default(D, L, Body), \quad (13)$$

$$holds(\neg L).$$

$$defeated(D) \leftarrow default(D, L, Body), \quad (14)$$

$$default(D_1, L_1, Body_1),$$

$$prefer(D_1, D),$$

$$hold(Body_1),$$

$$not\ defeated(D_1).$$

This collection of axioms is different from the original one presented in [5]:

1. we do not distinguish between *holds* and *holds_by_default*, since our goal is to use prioritized default theories in reasoning about actions; in this context it is not interesting to know whether a fluent is made true by an action or by inertia¹.
2. in rule (14) we do not require D and D_1 to be conflicting defaults.

3 Action Theories as Prioritized Default Theories

We will show now how each action theory can be represented by a prioritized default theory, whose semantics coincides with the action theory's entailment relation. The language to represent an action theory (D, I) in prioritized default theories consists of

¹ There are other approaches in reasoning about actions that do emphasize this point, but we are not interested in this distinction at this point in time.

- atoms of the form $f(T)$ (f is true at the time moment T),
- atoms of the form $possible(a, T)$ (a is executable at the time moment T),
- atoms of the form $occ(A, T)$ (action A occurs at the time moment T),
- rule names of the form $dynamic(f, a, T)$,
- rule names of the form $causal(f, T)$,
- rule names of the form $executable(a, T)$,
- default names of the form $inertial(f, T)$,

where f is a fluent literal, a is an action, and T is an integer representing units of time along a history. The translation of an action theory (D, Γ) into a prioritized theory $\Pi(D, \Gamma)$ is as follows²:

- For each dynamic law “ a **causes** f **if** p_1, \dots, p_n ” in D , $\Pi(D, \Gamma)$ contains the rules

$$rule(dynamic(f, a, T), f(T+1), [p_1(T), \dots, p_n(T), possible(a, T)]) \leftarrow occ(a, T) \quad (15)$$

- For each executability condition “ a **executable if** q_1, \dots, q_m ”, $\Pi(D, \Gamma)$ contains

$$rule(executable(a, T), possible(a, T), [q_1(T), \dots, q_m(T)]) \quad (16)$$

- For each static causal law “ f **if** p_1, \dots, p_m ”, $\Pi(D, \Gamma)$ contains the set of rules

$$rule(causal(f, T), f(T), [p_1(T), \dots, p_n(T)]) \quad (17)$$

- The inertial axiom is represented by the set of defaults

$$default(inertial(f, a, T), f(T+1), [f(T)]) \quad (18)$$

where f is a fluent literal, a is an action, and T is a sequence of actions.

- Finally, for each axiom “**initially** f ” in Γ , $\Pi(D, \Gamma)$ contains the fact

$$holds(f(0)). \quad (19)$$

For each action theory (D, Γ) , let $\Pi^k(D, \Gamma)$ be the logic program consisting of (i) The set of independent rules (9)-(14); (ii) The set of rules (15)-(18) in which T ranges between 0 and n , and (iii) The set of facts of the form (19). In the next two theorems, we prove the correctness of $\Pi^k(D, \Gamma)$. That is, we prove that the semantics provided by the prioritized default theory coincides with the action theory semantics. Let M be an answer set of $\Pi^k(D, \Gamma)$. Let $s_i(M) = \{f \mid holds(f(i)) \in M\}$. We begin with the soundness of $\Pi^k(D, \Gamma)$.

Theorem 1. *Let (D, Γ) be a complete and consistent action theory. Then, for every sequence of actions a_1, \dots, a_k such that there exists a trajectory $s_0 a_1 s_1 \dots a_k s_k$,*

- *for every answer set M of*

$$\Pi^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \dots, k\},$$

$$s_{i+1}(M) \in \Phi(a, s_i(M)) \text{ for every } i, 0 \leq i \leq k-1;$$

- *for every trajectory $s_0 a_1 s'_1 \dots a_k s'_k$ there exists an answer set M of*

$$\Pi^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \dots, k\},$$

$$\text{such that } s_i(M) = s'_i \text{ for every } i, 1 \leq i \leq k.$$

² In all of the rules and defaults, T is an integer representing time units.

4 Computing the Entailment Relation \models

The program $\Pi(D, \Gamma)$ can be implemented using **smodels** [14]. To make this possible, we need to introduce a collection of predicates to overcome the limitations of the input language; in particular, **smodels** does not support the list operator and requires finite domains and domain predicates to perform grounding.

Let us call the **smodels** program $SM(D, \Gamma)$. It is similar to the implementations of answer set planning proposed in the literature [8, 18]. It makes use of a time variable representing time instants along a trajectory. In all rules described below, we denote the time variable with T . The program $SM(D, \Gamma)$ makes use of the following predicates³:

- for each action a and for each fluent f we introduce the facts $action(a)$ and $fluent(f)$.
- each rule of the form

$$rule(dynamic(f, a, T), f(T+1), [p_1(T), \dots, p_n(T), possible(a, T)]) \leftarrow occ(a, T).$$

is encoded as the **smodels** rule

$$rule(dynamic(f, a, T), true(f, T + 1), precondition_set(a, T)) \leftarrow occ(a, T). \quad (20)$$

where the predicate $true(f, T)$ represents the truth value of the fluent literal f at time T (if $holds(true(f, T))$ is true, then f is true at time T). The predicate $precondition_set(a, T)$ is defined by the following rules:

$$\begin{aligned} &in(true(p_1, T), precondition_set(a, T)). \\ &\dots \\ &in(true(p_n, T), precondition_set(a, T)). \\ &in(true(possible(a, T)), precondition_set(a, T)). \end{aligned}$$

Similar encoding is used for the rules of the form (16)-(18).

- The two rules (11)-(12) are replaced by the rules

$$set(nil) \leftarrow \quad (21)$$

$$not_holds_set(S) \leftarrow set(S), in(F, S), not\ holds(F). \quad (22)$$

$$holds_set(S) \leftarrow set(S), not\ not_holds_set(S). \quad (23)$$

These rules state when a set (the body of a rule or a default is true). The first rule defined the constant nil representing the empty set.

- The predicate $name$ is used to define the names of rules and defaults, e.g., $name(dynamic(a, f, T))$ is introduced for the dynamic law “ a causes f if p_1, \dots, p_n .”
- The predicate $contrary$, defined by the rule $contrary(true(L, T), true(\neg L, T))$, is introduced to derive the negation of a literal in the prioritized default theory language. This is used to facilitate the application of defaults and rules, i.e., to implement the rules (10), (13), and (14).

³ **smodels** codes for some examples and the domain independent rules can be found at www.cs.nmsu.edu/~tson/preferences.

Let (D, Γ) be a complete and consistent action theory, and let $\alpha = a_1, \dots, a_n$ be a sequence of actions. With $SM^n(D, \Gamma)$ we denote the above **smodels**-rules for (D, Γ) in which the time variable ranges from 0 to n . With $SM^\alpha(D, \Gamma)$ we denote the program containing $SM^{|\alpha|}(D, \Gamma)$ and the set of action occurrences $\{occ(a_1, 0), \dots, occ(a_n, n-1)\}$. Following the result from [10, 18] we can show that $SM^\alpha(D, \Gamma)$ and $\Pi^\alpha(D, \Gamma)$ have “equivalent” answer sets—each answer set of $SM^\alpha(D, \Gamma)$ can be converted into an answer set of $\Pi^\alpha(D, \Gamma)$ and vice versa.

5 Finding a Trajectory using $SM^n(D, \Gamma)$

The discussion in the previous section shows that $SM^n(D, \Gamma)$ can be used to compute the entailment relation of (D, Γ) . In this section, we discuss the use of $SM^n(D, \Gamma)$ in finding a trajectory $s_0 a_1 \dots a_n s_n$ that satisfies the following properties:

1. $s_n \models \varphi$ for some given fluent formula φ —this means that the trajectory is a possible plan to accomplish the goal φ ;
2. the trajectory $s_0 a_1 \dots a_n s_n$ satisfies some soft constraints that are expressed as preferences between actions, between fluent formula, or between the trajectories.

5.1 Finding A Trajectory for φ

Let (D, Γ) be an action theory and φ be a conjunction of fluent literals $f_1 \wedge \dots \wedge f_k$ ⁴. We are interested in finding a trajectory $s_0 a_0 \dots a_n s_n$ for φ . As it is customary in answer set planning, we add to $SM^n(D, \Gamma)$ the set of rules to generate action occurrences and to represent the goal. This set of rules consists of:

$$goal(T) \leftarrow time(T), holds(true(f_1, T)), \quad (24)$$

...

$$holds(true(f_k, T)).$$

$$\leftarrow not\ goal(length). \quad (25)$$

$$1\{occ(A, T) : action(A)\}1 \leftarrow time(T), T < length. \quad (26)$$

Let $SM^{Plan,n}(D, \Gamma, \varphi)$ be the program consisting of the rules of $SM^n(D, \Gamma)$ and the set of rules (24)-(26), in which time variable takes values from 0 to n . It is easy to see that the following holds:

1. If $s_0 a_1 \dots a_n s_n$ is a trajectory for φ then $SM^{Plan,n}(D, \Gamma, \varphi)$ has an answer set S ⁵ such that
 - (a) $occ(a_i, i-1) \in S$ for every integer $i, 1 \leq i \leq n$, and
 - (b) $s_i = \{f \mid holds(true(f, i)) \in S\}$.
2. If $SM^{Plan,n}(D, \Gamma, \varphi)$ has an answer set S such that
 - (a) $occ(a_i, i-1) \in S$ for every integer $i, 1 \leq i \leq n$, and
 - (b) $s_i = \{f \mid holds(true(f, i)) \in S\}$
then $s_0 a_1 \dots a_n s_n$ is a trajectory for φ .

⁴ Fluent formula can be dealt with as in [18].

⁵ Observe that S is actually a stable model.

5.2 Finding a Preferred A Trajectory

A trajectory for a fluent formula φ is a possible plan to achieve φ . In many situations, it is desirable to find one, among several possible trajectories, that satisfies certain constraints. For example, *riding a bus* and *taxi* are two alternatives to go to the airport. An agent might choose to take the bus because he does not like taxi drivers. But he is willing to take the taxi if the bus does not run. Here, the agent has a preference between the actions he can execute and he would like to choose the trajectory that suits him best. We will call this a preference between trajectories and discuss how we can generate trajectories satisfying an agent's preferences using $SM^{Plan,n}(D, \Gamma, \varphi)$. In the process, we extend the prioritized default theory for representing the preferences between rules and literals. We will discuss preferences between actions, formulas, and trajectories.

Preferences Between Rules. To say that we do not prefer a rule r means that we do not want to use r . It does not necessarily mean that r cannot be applied, but it means that if r can be replaced, then we should do so. For this reason, we use literals of the form

$$block(r, [l_1, \dots, l_m]); \quad (27)$$

to describe conditions under which a rule r should not be used. In particular, literals of this type can be used to represent preferences between the rules. For example, if we want to express the fact that if two rules r_1, r_2 have the same consequence and we prefer to use r_2 instead of r_1 , then we can write $block(r_1, body(r_2))$.

To implement the new type of rules in prioritized default theory, we replace rule (9) by the following rule:

$$holds(L) \leftarrow rule(R, L, Body), hold(Body), not\ blocked(R). \quad (28)$$

and add the next rule to $\Pi(D, \Gamma)$:

$$blocked(R) \leftarrow block(R, Body), hold(Body). \quad (29)$$

This is used to block the application of the rule R . Observe that blocking a rule is different than defeating a default; a rule can be blocked only at the explicit will of the domain specifier, while a default can be defeated if its application introduces inconsistencies.

We now show that this modification allows us to deal with preferences between actions. We assume that we have an irreflexive partial order between actions, $prefer(a, b)$, to represent the preferences between actions. Intuitively, this means that action a is preferred to action b and we would like to consider all the trajectories containing a in the place of b before considering those containing b . More precisely:

Definition 1 (Preferred Trajectory). A trajectory $\alpha = s_0 a_1 s_1 \dots, a_n s_n$ is said to be preferred to a trajectory $\beta = s_0 b_1 s'_1 \dots, b_n s'_n$ with respect to a set of preferences P , denoted by $\alpha \prec_P \beta$, if

1. there exists an integer i , $1 \leq i \leq n$, such that $prefer(a_i, b_i) \in P$, and
2. for every integer j , $1 \leq j < i$, $prefer(b_j, a_j) \notin P$.

Definition 2 (Most Preferred Trajectory). A trajectory $\alpha = a_1, \dots, a_n$ is said to be a most preferred trajectory with respect to a set of preferences P if there exists no trajectory β such that $\beta \prec_P \alpha$.

Remark 1. \prec_P is an antisymmetric, transitive, and irreflexive relation.

Let us now encode the preferences between actions as rules of $SM^{Plan,n}(D, \Gamma)$. For simplicity, instead of translating the set of preferences between actions into literals of the form (27) we will encode it directly into logic programming. For each $prefer(a, b)$, and for each dynamic causal law “ a causes g if p_1, \dots, p_m ”, we define a rule

$$\begin{aligned} &block(dynamic(F, b, T), [true(p_1, T), \dots, true(p_m, T), possible(a, T)]) \\ &\leftarrow goal(length). \end{aligned} \quad (30)$$

Intuitively, this rule says that if (at the time moment T) it is possible to execute the action a and the goal is achievable, then action b should not be executed. Let (D, Γ) be an action theory and P be a set of preferences on actions in D . Let $SM^{Pref,n}(D, \Gamma, \varphi)$ be the program consisting of (1) the program $SM^{Plan,n}(D, \Gamma, \varphi)$, and (2) the set of rules (30) with the time variable ranging between 0 and n . It is easy to see that, when $prefer(a, b)$ is present and both actions are executable and lead to the goal, then a will be used first. However, the rules of the form (30) do not warrant completeness, i.e., it does not guarantee that a trajectory is found if the problem has a solution.⁶

On the other hand, if we use the **smodels** construct **maximize**, the most preferred trajectory can always be found, by adding the following optimal rule to the program $SM^{Pref,n}(D, \Gamma, \varphi)$: for each $prefer(a, b)$ and for each time point t

$$\mathbf{maximize}[occ(a, t) = 1, occ(b, t) = 0].$$

In our future work, we plan to investigate whether a sound and complete implementation in the prioritized default theory can be realized.

Preferences between Formula. The second type of preferences that we consider is between formulae. Unlike preferences between rules, this type of preference is often a soft constraint or a secondary goal that an agent has in mind when selecting a trajectory for his goal. Consider for example our agent in the previous section. He might prefer to take the bus because it is cheap (*save.money*). Here, the primary goal of the agent is to be at the airport and his soft constraint is to save money. The trivial choice would be to take the bus. Taking a taxi would be used as the last resource. This type of preference can be added to a prioritized default theory by introducing preferences of the form

$$\varphi_1 \prec \varphi_2. \quad (31)$$

Again, we assume that \prec is an irreflexive partial order. Thus, for a finite set of preferences of the form (31), there exists a finite number of maximal length sequences of formulae $\varphi_1 \prec \dots \prec \varphi_p$. To implement (31), for each maximal sequence $\varphi_1 \prec \dots \prec \varphi_p$, we add the optimal rule [13]

$$\mathbf{maximize}[\varphi_1 = 0, \dots, \varphi_p = p] \quad (32)$$

to $SM^{Plan,n}(D, \Gamma)$. We are assuming that the computation of the answer sets maximizes each rule of type (32). Observe that the current implementation of **smodels** does

⁶ For instance, when two actions a and b are possible, and a is preferred to b but does not lead to the final goal, then the program may fail to produce a trajectory.

not guarantee this behavior⁷. If we want to use the current version of **smodels**, then we need to additionally require that the preference relation \prec is total order.

General Preferences Between Trajectories. In general, an agent can have several preferences on a trajectory. For example, he might prefer to use an action a over an action b , he might also prefer that whenever he has to execute an action c then d should be the next action, etc. It has been discussed in [1] that many preferences or constraint of this type can be conveniently expressed as a temporal logic formula. Since the truth value of a temporal logic formula can be easily checked given a trajectory, this feature can be added to our framework by

- adding rules for checking the truth value of temporal logic formulae, that associate each temporal logic formula, say φ , to a new boolean variable φ^T , whose truth value in the final state corresponds to the satisfiability of φ w.r.t. the chosen trajectory (as illustrated in [18]),
- adding an optimal rule

maximize $[\varphi^T = 1, \text{not } \varphi^T = 0]$

to the program $SM^{Plan,n}(D, I)$, that allows us to find trajectories satisfying φ before considering those not satisfying it.

6 Conclusions

In this paper we present a formalism for reasoning about actions in the context of prioritized default theory. In this process, we provide an encoding of an action theory in prioritized default theory, whose answer set semantics coincides with the entailment relation of the action theory. In addition, prioritized default theory are very expressive and can be used to model dynamic domains that cannot be expressed using, e.g., the language \mathcal{B} ; for example

- domains with non-inertial fluents (e.g., a spring-loaded door is open immediately after the push action is performed, but it will automatically revert to close at the next moment of time).
- domains with exogenous actions (e.g., a domain where a driver agent stops at the traffic light, and expects the light to change color; i.e., the driver agent expects the change color action to occur (exogenously)).

Our focus in this work is on using prioritized default theory to encode different forms of preferences between trajectories. In the process, we extend prioritized default theory to allow preferences between rules and formulae. We show how these features can be easily implemented in answer set programming. We also employ these new features to directly express three different types of preferences between trajectories, i.e., preferences between actions, between final states, and general preferences between trajectories.

The preliminary experiments performed have provided encouraging results, and work is in progress to establish the full range of capabilities of this approach. In particular, we intend to use the proposed framework in the design of bioinformatics applications—i.e., software agents in charge of mapping high-level biological process descriptions

⁷ Currently **smodels** maximizes only the last optimal rule in the program.

into a predefined collection of software services [16]—and in the development of Web accessibility agents for visually impaired individuals [17].

Several other approaches to dealing with preferences between logic programming rules have been proposed.⁸ In our future work we plan to investigate the use of these methods in representing and reasoning with preferences among actions.

Acknowledgments: The authors wish to thank M. Gelfond and the anonymous referees for the comments on various drafts of this work. Research has been supported by NSF grants EIA-0130887, CCR-9875279, HRD-9906130, and EIA-9810732 and by a grant from NASA.

References

1. F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1,2):123–191, 2000.
2. R. Fikes and N. Nilson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.
3. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programs. In *ILPS*, MIT Press, 1988.
4. M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.
5. M. Gelfond and T.C. Son. Prioritized default theory. In *Selected Papers from the Workshop on Logic Programming and Knowledge Representation*, pages 164–223. Springer, 1998.
6. M. Ginsberg and D. Smith. Reasoning about actions I. *Artificial Intelligence*, 35, 1988.
7. R. Kowalski and M. Sergot. A logic-based calculus of events. *NGC*, 4:67–95, 1986.
8. V. Lifschitz. Answer set planning. In *Int. Conf. Logic Programming*, pages 23–37, 1999.
9. V. Lifschitz and H. Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proceedings of the Eleventh International Conf. on Logic Programming*, pages 23–38, 1994.
10. V. Lifschitz and H. Turner. Representing transition systems by logic programs. In *Proceedings Int. Conf. on Logic Programming and Nonmonotonic Reasoning*, pages 92–106, 1999.
11. J. McCarthy. Epistemological problems of artificial intelligence. In *Proceedings Int. Joint Conference on Artificial Intelligence*, pages 1038–1044. 1977.
12. J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502. 1969.
13. I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.
14. I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Procs. LPNMR*, pages 420–429, 1997.
15. E. Pednault. ADL and the state-transition model of actions. *Journal of Logic and Computation*, 4(5):467–513, October 1994.
16. E. Pontelli, G. Gupta, D. Ranjan, and B. Milligan. A Domain Specific Language for Solving Phylogenetic Inference Problems. TR-CS-001/2002, New Mexico State University, 2002.
17. E. Pontelli and T. Son. Navigating HTML Tables: Planning, Reasoning, and Agents. In *Int. Conference on Assistive Technologies*. ACM Press, 2002.
18. T.C. Son, C. Baral, and S. McIlraith. Domain dependent knowledge in planning - an answer set planning approach. In *Procs. LPNMR*, pages 226–239, Vienna, 2001.
19. T.C. Son and E. Pontelli. Reasoning About Actions in Prioritized Default Theory. TR-CS-002/002, New Mexico State U., 2002.
20. H. Turner. Representing actions in logic programs and default theories. *Journal of Logic Programming*, 31(1-3):245–298, May 1997.

⁸ References to these proposals are omitted due to lack of space.