Reasoning about Actions and Planning with Preferences Using Prioritized Default Theory

Tran Cao Son and Enrico Pontelli
Knowledge Representation, Logic, and Advanced Programming Laboratory
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
{tson,epontell}@cs.nmsu.edu

October 14, 2003

Abstract

This paper shows how action theories, expressed in an extended version of 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*. This extension provides a natural framework to introduce different types of preferences in action theories—*preferences between actions* and *preferences between final states*. In particular, we demonstrate how these preferences can be expressed within extended prioritized default theory. We also discuss how this framework can be implemented in terms of answer set programming.

1 Introduction

One of the central aspects in formalizing commonsense reasoning is represented by reasoning about *actions* and their *effects* (*Reasoning about Action and Change (RAC)*). Research in RAC has mostly focused on developing *formalisms* for representing and reasoning about actions and their effects. Dynamic domains can be conveniently described using specialized languages—e.g., situation calculus [22], event calculus [14], STRIPS [7], and action description languages [26, 10]. In these languages, a dynamic domain is represented as an *action theory*, where effects of actions are encoded as sets of propositions. The semantics of an action theory is defined by an entailment relation that determines what will be true/false after an action sequence is executed starting from a given initial state.

For several years, the *frame problem* [22], the *ramification problem* [12], and the *qualification problem* [21] have been at the center of RAC's research. Intuitively, the frame problem is the problem of concisely describing 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 the relationship between fluents). The qualification problem 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, such as the situation calculus [30], high-level action description languages [10, 20], and event calculus [31].

The strong connection between RAC and default reasoning has been discussed in [16], where it is shown that the law of inertia can be viewed as a default. As such, it is natural to think that any logical framework for default reasoning could be a suitable framework to support RAC as well. Indeed, this is the approach explored in [36] where Reiter's default logic [29] is used.

Despite the strong connection between RAC and default reasoning, it is interesting to observe that several important problems in default reasoning—such as the problem of preferring a conclusion over others in presence of conflicts—do not seem to arise in RAC. Can we attribute it to the fact that action theories are often assumed to be consistent and there is only one default (the law of inertia) which will be overridden whenever conflicts arise? To a certain extent, this is true in the context of RAC, where the focus has been limited to predicting the effects of actions or action sequences. On the other hand, this is not the case in *planning with preferences* (or constraints), where the goal is find a trajectory that achieves a predefined goal and at the same time satisfies certain preferences.

Current approaches to RAC provide the ability to construct trajectories achieving a predefined goal. Nevertheless, in many situations, it is desirable to find one among several possible trajectories that satisfies certain constraints. For example, when developing strategies to obtain a loan to purchase a house, a user may have preferences towards local lenders, at parity of conditions. These preferences can be viewed as *soft constraints* on a trajectory or a plan, that may or may not be satisfied depending on the particular situation. In the context of planning, this could be also viewed as an indication of the plan quality [23] or plan with optimal utility [13].

In this paper we show that *Prioritized Default Theory* [11] 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 can be elegantly translated into semantically equivalent prioritized default theories.[1] The novel encoding proposed in this paper provides an elegant and concise way to describe actions and their effects, along with effective solutions to the frame, ramification, and qualification problems.

We also explore ways to allow the action language to encode different types of *user's preferences*, and use them to guide the development of *preferred* plans. The preferences allow the user to express a bias towards certain types of trajectories to achieve the given goal—i.e., certain trajectories will be *preferred* to others. In this work we explore alternative forms of preferences at the level of the action language:

1. Preferences between *actions*—i.e., the ability to define an order of preference between actions to be used in developing a plan; e.g.,

   prefer **take a limo** *to* **drive the car**

2. Preferences between *final states*—although all final states reached by valid trajectories are guaranteed to satisfy the required goal, the user may be interested in suggesting additional criteria to choose final states; e.g.,

[1] In this work we concentrate on the action language $\mathcal{B}$ [10].

prefer to have more than $10,000 left at the end of the trip.

We demonstrate how these different forms of preferences can be elegantly encoded within a generalized prioritized default theory framework, where 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).

Finding a preferred trajectory that achieves a certain goal is not new in planning. Approaches to planning have tried to address this issue by using a utility function [13] or developing specific algorithms for planning in the presence of preferences [23]. The key idea in using a utility function is that it will allow us to find the trajectories with minimal (maximal) cost. This approach is very general. The main disadvantage of this approach is that coming up with a good utility function is not easy in many cases.

To the best of our knowledge, the only work addressing this issue in the context of logic programming based approaches to planning is from [6] and in an early version of this work [35]. In [6], each action is assigned a cost and the minimal cost plan is searched. In contrary, we emphasize the use of prioritized default theory in expressing the preferences between actions and formulae. Since domain-dependent knowledge could be viewed as preferences on trajectories, an early work by one of the authors [34] could also be viewed as an attempt to add different forms of preferences to answer set planning. In this paper, we emphasize on allowing users to specify their preferences.

The paper is organized as follows. Section 2 describes the syntax and semantics of the action language $\mathcal{B}^d$ used in this work—the language is a novel extension of the language $\mathcal{B}$ with non-inertial fluents. Section 3 provides a brief overview of prioritized default theories. Section 4 describes our proposed translation from the action language $\mathcal{B}^d$ to prioritized default theories. Section 5 describes how preferences can be expressed, encoded, and used to guide the process of computing trajectories. Section 6 demonstrates how trajectories can be effectively computed using an answer set solver, such as SMODELS. Finally, Section 7 provides some concluding remarks.

2 The Action Language $\mathcal{B}^d$

In this section, we present the basic action language that we will employ to encode action theories. The language, called $\mathcal{B}^d$, is an extension of the the language $\mathcal{B}$ [10], with the addition of default knowledge about *non-inertial fluents*.

2.1 Syntax

$\mathcal{B}^d$ is a language schema that allows the development of specific action description languages; each specific action description language includes

- a non-empty set of symbols **F**, called *fluent names*;
- a non-empty set of symbols **A**, called *action names*.

Fluents are used to describe *properties*, whose value depends on the current state of the world. The set of fluent names $\mathbf{F}$ is partitioned in two subsets, $\mathbf{F} = \mathbf{F_I} \cup \mathbf{F_N}$, with $\mathbf{F_I} \cap \mathbf{F_N} = \emptyset$. The fluents in $\mathbf{F_I}$ are called *inertial fluents*, while those in $\mathbf{F_N}$ are called *non-inertial fluents*. Intuitively, non-inertial fluents encode properties that are exempt from the commonsense laws of inertia.

2.2 Formulae and Action Descriptions

A *fluent literal* is a fluent name, possibly preceded by $\neg$. A *fluent formula* is a propositional combination of fluent literals.

There are four type of propositions in $\mathcal{B}^d$:

- *Dynamic Causal Law:*
$$a \quad \textbf{causes} \quad f \ \textbf{if}\, p_1, \ldots, p_n \tag{1}$$
where a is an action name ($a \in \mathbf{A}$), f is a fluent literal, and $p_1, \ldots, p_n$ are fluent literals.

- *Static Causal Law:*
$$f \quad \textbf{if} \quad p_1, \ldots, p_m \tag{2}$$
where f is an inertial fluent literal and $p_1, \ldots, p_m$ are fluent literals.

- *Executability Conditions:*
$$a \quad \textbf{executable\_if} \quad p_1, \ldots, p_n \tag{3}$$
where a is an action name and $p_1, \ldots, p_n$ are fluent literals.

- *Default Knowledge:*
$$g \quad \textbf{by\_default} \tag{4}$$
where g is a non-inertial fluent literal—i.e., a fluent literal constructed using a fluent $f \in \mathbf{F_N}$.

The dynamic law (1) represents the (conditional) effect of action a on the fluent literal f, while the causal law (2) states a causal relationship between the fluent literals f and $p_1, \ldots, p_m$. The proposition (3) represents the conditions under which the action a can be executed. [2] Propositions of the form (4) represent our default knowledge about non-inertial fluents—by asserting what is the default value for the fluent g.

A *domain description* (or *domain*) is a set of propositions of the forms (1)-(4), with the additional restrictions:

- for each $f \in \mathbf{F_N}$, exactly one of the following propositions is present in the domain description:
$$f \ \textbf{by\_default} \qquad \neg f \ \textbf{by\_default}$$

[2]Observe that the **if** part of rule (1) is used to condition the outcome of the action, not to determine the executability of the action, as in rule (3).

- non-inertial fluents are changed only by direct effects of actions, i.e., non-inertial fluents cannot occur on the left hand side of static causal laws.

Axioms in $\mathcal{B}^d$ are propositions of the form

$$\textbf{initially } f \tag{5}$$

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

$$\varphi \textbf{ after } \alpha \tag{6}$$

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 initial state.[3]

An *action theory* is a pair (D, Γ), where D is a domain description and Γ is a collection of propositions of type (5) (the *initial state*).

2.3 An Example

The next example illustrates the use of the language $\mathcal{B}^d$ to describe a simple dynamic domain. Consider the problem of finding ways to exit a building. The building has two exits, a front door and a back door. Both doors require a key to open them; the key for the front door is available only from the janitor, while all employees have a key to the back door. Opening the back door will cause an alarm to go off (as long as the door is open) and the police will be automatically notified. Furthermore, the back door has a spring system that will automatically close it when released.

2.3.1 Fluents and Actions

The domain description makes use of the fluents:

$at(X)$	*the person is at location X*
$opendoor(Y)$	*the door Y is open*
$haveKey(Y)$	*the person has the key to the door Y*
$inside$	*the person is inside the building*
$alarm$	*the alarm is sounding*
$police\_alerted$	*the police has been notified*

In the fluent schemas above, X represents a possible location (i.e., *office*, *frontdoor*, *backdoor*) while Y indicates one of the doors (i.e., either *frontdoor* or *backdoor*). In particular, all the fluents, except for the $opendoor(backdoor)$, belong to $\mathbf{F_I}$, while the fluent $opendoor(backdoor)$ is non-inertial.

[3]We could also generalize the discussion and allow this query to be posed w.r.t. an arbitrary state.

The set of action names **A** includes the following elements:

$goto(X)$	*the person moves to location* X
$open(Y)$	*the person opens the door* Y
$cross(Y)$	*the person crosses the door* Y
$getKey(backdoor)$	*get key for the back door from the janitor*

In this context, X represents a possible location (i.e., $frontdoor$, $backdoor$, or *office*), while Y represents one of the doors ($frontdoor$ or $backdoor$).

2.3.2 Domain Description

The domain description includes the following set of dynamic causal laws, describing the effect of the actions:

$goto(X)$	**causes**	$at(X)$		
$open(Y)$	**causes**	$opendoor(Y)$	**if**	$at(Y)$
$cross(Y)$	**causes**	$\neg inside$	**if**	$inside, at(Y)$
$cross(Y)$	**causes**	$inside$	**if**	$\neg inside, at(Y)$
$getKey(backdoor)$	**causes**	$haveKey(backdoor)$		

The actions have some preconditions that have to be met in order to allow their execution:

$goto($*office*$)$	**executable_if**	$inside$
$open(Y)$	**executable_if**	$haveKey(Y)$
$cross(Y)$	**executable_if**	$at(Y), opendoor(Y)$
$getKey(backdoor)$	**executable_if**	$true$

As mentioned earlier, the backdoor has a spring system that will force it to close immediately after it has been opened. This is modeled as a non-inertial fluent with a default value expressing the fact that the door is closed:

$\neg opendoor(backdoor)$ **by_default**

The causal relationship between fluents is described through a set of static causal laws[4]:

$\neg at(X)$	**if**	$at(Y), X \neq Y$
$alarm$	**if**	$open(backdoor)$
$police\_notified$	**if**	$open(backdoor)$
$\neg alarm$	**if**	$\neg open(backdoor)$
$\neg at(office)$	**if**	$\neg inside$
$inside$	**if**	$at(office)$

[4]Notice that the effects of some static causal laws can also be described using dynamic laws. For example, $goto(X)$ **causes** $at(X)$ and $goto(X)$ **causes** $at(Y)$ for $Y \neq X$. Deciding whether to use dynamic law or static law to represent certain knowledge of the domain is a challenging task that deserves a careful investigation and is outside the scope of this paper.

2.4 Semantics

The semantics of a domain description D in $\mathcal{B}^d$ is defined by a corresponding *transition function* Φ_D. For the sake of readability, we will omit D from Φ_D whenever the domain description is clear from the context. The transition function is aimed at describing the possible states of the world ($\Phi_D(a, s)$) an agent might be in after she/he executes the action a in the state s. When $\Phi_D(a, s)$ is the empty set, then this means that the action a is not executable in s.

2.4.1 States

Let D be a domain description in $\mathcal{B}^d$. An *interpretation* I of the fluents in D is a maximal consistent set of fluent literals from **F**. A fluent f is said to be true (respectively false) in I iff $f \in I$ (respectively $\neg f \in I$). The truth value of a fluent formula in I is defined recursively over the propositional connectives in the usual way. For example, give two fluent formulae φ and ψ, the fluent formula $\varphi \wedge \psi$ is true in I iff φ is true in I and ψ 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 let K be a set of static causal laws. We say that U is *closed under* K if, for every static causal law

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

in K, whenever $\{p_1, \ldots, p_n\} \subseteq U$ we have that $f \in U$. By $Cl_K(U)$ we denote the least consistent set of fluent literals from F that contains U and is closed under K.

A *state* s of D is an interpretation of the fluents in **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

$$a \textbf{ executable\_if } f_1, \ldots, f_n$$

in D such that $s \models f_1 \wedge \ldots \wedge f_n$. If the executability condition

$$a \textbf{ executable\_if } true$$

belongs to D, then a is executable in every state of D.

2.4.2 The Transition Function

The *immediate effect of an action a* in state s is the set

$$E(a, s) = \{f \mid \ [\, a \textbf{ causes } f \textbf{ if } f_1, \ldots, f_n \,] \in D, \ \ s \models f_1 \wedge \ldots \wedge f_n\}$$

Additionally, given a domain description D, let us introduce the set

$$Def(D) = \{f \ \mid \ [\, f \textbf{ by\_default } \,] \in D \ \}$$

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

$$\Phi_D(a, s) = \{s' \ \mid \ s' \text{ is a state and } s' = Cl_{D_C}(E(a, s) \cup (s \cap s' \cap (F_I \cup \overline{F_I})) \cup (Def(D) \setminus E(a, s)))\}$$

where D_C is the set of static causal laws in D and $\overline{F_I} = \{\neg f \mid f \in F_I\}$. If a is not executable in s, then $\Phi_D(a, s) = \emptyset$. For each domain description D in $\mathcal{B}^d$, the transition function Φ_D is unique. Since $Cl_{D_C}(X)$ could be empty, it is possible that $\Phi(a, s) = \emptyset$ even when a is executable in s. When this happens, we say that D is *inconsistent*. This situation is not particularly realistic, as the execution of an action a in the state s should result in another state whenever a is executable in s. In other words, this anomaly is an indication of a possible mistake in the domain description. As out interest in this paper is to use action theories in planning and not the study of RAC per se, we limit ourselves on domains without this problem. In other words, we will assume that domain descriptions in this paper are *consistent*, i.e., domain descriptions in which $\Phi_D(a, s) \neq \emptyset$ for every action a and state s, if a is executable in s.

With a slight abuse of notation, we will also extend the transition function to operate on *sequences of actions*. Given a sequence of actions $a_1 \cdots a_n$ ($n \geq 0$)

$$\Phi_D(a_1 \cdots a_n, s) = \begin{cases} \{s\} & \text{if } n = 0 \\ \bigcup_{s' \in \Phi_D(a_1 \cdots a_{n-1}, s)} \Phi_D(a_n, s') & \text{otherwise} \end{cases}$$

Given a domain D with transition Φ_D, a sequence $s_0 a_1 s_1 \ldots a_n s_n$, where s_i's are states and a_i's are actions, is called a *trajectory* in D if

$$s_{i+1} \in \Phi_D(a_{i+1}, s_i) \text{ for every } i,\ 0 \leq i \leq n-1.$$

A trajectory $s_0 a_1 s_1 \ldots a_n s_n$ is a trajectory of a fluent formula Δ if $s_n \models \Delta$.

An action theory (D, Γ) is consistent if D is consistent and

$$s_0 = \{f \mid \ [\ \textbf{initially } f\ ] \in \Gamma\}$$

is a state of D. s_0 is called the *initial state* of (D, Γ).

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. In this paper we opt to focus on complete action theories; the issues arising from incompleteness (e.g., sensing actions [33] and conformant planning [32]) are orthogonal to the scope of this work.

Finally, given an action theory (D, Γ) whose initial state is s_0, a fluent formula φ, and an action sequence $\alpha = a_1, \ldots, a_n$, we say that the query φ **after** α is entailed by (D, Γ), denoted by

$$(D, \Gamma) \models \varphi \textbf{ after } \alpha$$

if for every possible trajectory $s_0 a_1 s_1 \ldots a_n s_n$, φ holds in s_n ($s_n \models \varphi$). In what follows, we will consider only consistent and complete action theories.

Example 1 Let D_b be the domain description in Example 2.3 and Γ be the initial state containing the following propositions:

initially $at(\mathit{office})$
initially $\neg at(frontdoor)$
initially $\neg at(backdoor)$
initially $\neg open(frontdoor)$
initially $\neg open(backdoor)$
initially $\neg police\_notified$
initially $\neg alarm$
initially $haveKey(backdoor)$
initially $\neg haveKey(frontdoor)$
initially $inside$

The initial state s_0 in this action theory is

$$s_0 = \left\{ \begin{array}{l} at(\mathit{office}), \neg at(frontdoor), \neg at(backdoor), \\ haveKey(backdoor), \neg haveKey(frontdoor), inside, \\ \neg police\_notified, \neg alarm, \neg open(frontdoor), \\ \neg open(backdoor) \end{array} \right\}$$

The action $goto(backdoor)$ is executable in s_0. We can easily check that

$$s \in \Phi(goto(backdoor), s_0) \quad \Rightarrow \quad s \models at(backdoor)$$

This implies that $(D_b, \Gamma) \models at(backdoor)$ **after** $goto(backdoor)$. If we consider also the actions $opendoor(backdoor)$ and $cross(backdoor)$ then we can obtain

$$(D_b, \Gamma) \models \neg inside \textbf{ after } goto(backdoor), opendoor(backdoor), cross(backdoor).$$

3 Prioritized Default Theory

Prioritized default theory has been discussed in [11]. 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 [8]. 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** [25, 4]) to compute the entailment relation of prioritized default theory. In this paper we begin with the theory proposed by Gelfond and Son in [11]. We then extend it to deal with preferences between rules.

A prioritized default theory consists 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

$$rule(r, l_0, [l_1, \ldots, l_m]) \tag{7}$$
$$default(d, l_0, [l_1, \ldots, l_m]) \tag{8}$$
$$prefer(d_1, d_2) \tag{9}$$

where r is a rule name, d, d_1, d_2 are default names, $l_0, \ldots, l_m$ are literals of the language $\mathcal{L}$, and $[\,]$ is the list operator. For convenience, we will refer to the atoms of the form (7), (8), and (9) as rules, defaults, and preferences, respectively. For a rule r, let $body(r)$ denote the list $[l_1, \ldots, l_m]$ and let $head(r)$ denote the literal l_0. Similar notation will be used for defaults. We assume that default names and rule names belong to two disjoint sets. The semantics of a default theory T is defined by the answer set semantics of a logic program [8], consisting of T and the following set of domain independent axioms:

- **Rules for Inference:**

$$holds(L) \leftarrow rule(R, L, Body), hold(Body). \tag{10}$$
$$holds(L) \leftarrow default(D, L, Body), hold(Body), not\ defeated(D). \tag{11}$$
$$hold([\,]) \leftarrow \tag{12}$$
$$hold([H|T]) \leftarrow holds(H), hold(T). \tag{13}$$

- **Rules for Defeating Defaults:**

$$defeated(D) \leftarrow default(D, L, Body), holds(L_1), contrary(L, L_1). \tag{14}$$
$$defeated(D) \leftarrow default(D, L, Body), default(D_1, L_1, Body_1), prefer(D_1, D), hold(Body_1), not\ defeated(D_1). \tag{15}$$

  where $contrary$ determines the opposite of a literal (e.g., $contrary(A, \neg A)$ holds for any atom A).

This collection of axioms, denoted by $\mathcal{P}$, is different from the original one presented in [11]:

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 inertia[5].

2. in rule (15) we do not require D and D_1 to be conflicting defaults.

[5]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.

4 Action Theories as Prioritized Default Theories

We will show now that each action theory can be elegantly represented by a prioritized default theory. The language for representing an action theory (D, Γ) in prioritized default theory consists of atoms of the form $f(\alpha)$ and $possible(a, \alpha)$, where f is a fluent literal, a is an action, and α is a sequence of actions. For convenience, we often use $|\alpha|$ to denote the length of α and α_i to denote the prefix of length i of α. $\alpha \sqsubseteq \beta$ denotes the fact that α is a prefix of β. $\circ$ is the concatenation operator between action sequences. The translation of an action theory (D, Γ) into a prioritized theory $\Pi(D, \Gamma)$ is performed as follows.

- For each dynamic law

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

  in D, $\Pi(D, \Gamma)$ contains the set of rules

  $$rule(dynamic(f, a, \beta), f(\beta \circ a), [p_1(\beta), \ldots, p_n(\beta), possible(a, \beta)]) \tag{16}$$

  where β is an arbitrary action sequence.

- For each executability condition

  $$a \textbf{ executable\_if } q_1, \ldots, q_m$$

  in D, $\Pi(D, \Gamma)$ contains the set of rules

  $$rule(executable(a, \beta), possible(a, \beta)), [q_1(\beta), \ldots, q_m(\beta)]) \tag{17}$$

  where β is an arbitrary action sequence.

- For each static causal law

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

  in D, $\Pi(D, I)$ contains the set of rules

  $$rule(causal(f, a, \beta), f(\beta \circ a), [p_1(\beta \circ a), \ldots, p_n(\beta \circ a), possible(a, \beta)]) \tag{18}$$

  where β is an arbitrary action sequence and a is an action.

- For each default law

  $$g \textbf{ by\_default}$$

  in D, $\Pi(D, I)$ contains the set of defaults

  $$default(def(g, a, \beta), g(\beta \circ a), [possible(a, \beta)]) \tag{19}$$

  where β is an arbitrary action sequence and a is an arbitrary action.

- The inertial axiom is represented by the set of defaults

  $$default(inertial(f, a, \beta), f(\beta \circ a), [f(\beta), possible(a, \beta)]) \tag{20}$$

  where f is an inertial fluent literal, a is an action, and β is an arbitrary sequence of actions.

- Finally, the set of axioms of the form (5) is represented by the set of facts

$$holds(f([])). \tag{21}$$

Notation: Let R_α denote the set of rules of the form (16)-(20) where $\beta \circ a \sqsubseteq \alpha$. Let R^k denote

$$R^k = \bigcup_{|\alpha| \leq k} R_\alpha$$

Let $\Pi^\alpha(D, \Gamma)$ denote the program consisting of the rules R_α, the set of rules (10)-(15), and the set of facts (21). For each integer k and for each action theory (D, Γ), let

$$\Pi^k(D, \Gamma) = \bigcup_{|\alpha| \leq k} \Pi^\alpha(D, \Gamma).$$

Example 2 *Let us consider the following simple action theory with actions a and b and fluents f and g where g is non-inertial:*

b	**causes**	f	**if**	$true$
a	**causes**	$\neg f$	**if**	g
a	**executable_if**	f		
b	**executable_if**	$true$		
g	**by_default**			
	initially	$\neg f$		
	initially	g		

For every action sequence β, $R_{\beta \circ a}$ consists of the following rules:

$$\begin{array}{l}
rule(executable(a,\beta), possible(a,\beta), [f(\beta)]) \\
rule(dynamic(\neg f, a, \beta), \neg f(\beta \circ a), [g(\beta), possible(a,\beta)]) \\
default(def(g, a, \beta), g(\beta \circ a), [possible(a,\beta)]) \\
default(inertial(f, a, \beta), f(\beta \circ a), [f(\beta), possible(a,\beta)]) \\
default(inertial(\neg f, a, \beta), \neg f(\beta \circ a), [\neg f(\beta), possible(a,\beta)]).
\end{array}$$

The set of rules $R_{\beta \circ b}$ is similar to $R_{\beta \circ a}$ and consists of the following rules and defaults:

$$\begin{array}{l}
rule(executable(b,\beta), possible(b,\beta), []) \\
rule(dynamic(f, b, \beta), f(\beta \circ b), [possible(b,\beta)]) \\
default(def(g, b, \beta), g(\beta \circ b), [possible(b,\beta)]) \\
default(inertial(f, b, \beta), f(\beta \circ b), [f(\beta), possible(b,\beta)]) \\
default(inertial(\neg f, b, \beta), \neg f(\beta \circ b), [\neg f(\beta), possible(b,\beta)])
\end{array}$$

and the set of rules of the form (21) consists of the following facts:

$$\begin{array}{l}
holds(\neg f([])) \\
holds(g([])).
\end{array}$$

We next discuss the properties of the program $\Pi^k(D, \Gamma)$. We will show that the action theories in $\mathcal{B}^d$ and their prioritized default theories are semantically equivalent.

Theorem 1 *For every consistent and complete action theory (D, Γ), the program $\Pi^k(D, \Gamma)$ is consistent.*

Proof. See Appendix A.

Given a program Π, let us denote with $lit(\Pi)$ the set of literals of Π. The following result holds:

Theorem 2 *Let (D, Γ) be a consistent and complete action theory and α be a sequence of actions with $|\alpha| < k$. Then,*

- *If M is an answer set of $\Pi^k(D, \Gamma)$ then $M^\alpha = M \cap lit(\Pi^\alpha(D, \Gamma))$ is an answer set of $\Pi^\alpha(D, \Gamma)$.*
- *If M^α is an answer set of $\Pi^\alpha(D, \Gamma)$ then there exists an answer set M of $\Pi^k(D, \Gamma)$ such that $M^\alpha = M \cap lit(\Pi^\alpha(D, \Gamma))$.*

Proof. See Appendix A.

In the next two theorems, we prove the correctness of $\Pi^k(D, \Gamma)$. In particular, we prove that the semantics provided by the prioritized default theory coincides with the semantics of the action theory. Let M be an answer set of $\Pi^k(D, \Gamma)$ and α be a sequence of actions with $|\alpha| \le k$. Let us define

$$s(\alpha, M) = \{f \mid holds(f(\alpha)) \in M\}.$$

We begin with the soundness of $\Pi^k(D, \Gamma)$.

Theorem 3 *Let (D, Γ) be a consistent and complete action theory, M be an answer set of $\Pi^k(D, \Gamma)$, α be a sequence of actions with $|\alpha| \le k$, and a be an action such that $s(\alpha, M) \neq \emptyset$ and $s(\alpha \circ a, M) \neq \emptyset$. Then,*

$$s(\alpha \circ a, M) \in \Phi(a, s(\alpha, M)).$$

Proof. See Appendix A.

The next theorem proves the completeness of $\Pi^k(D, \Gamma)$.

Theorem 4 *Let (D, Γ) be a consistent and complete action theory and $s_0 a_1 s_1 \ldots a_k s_k$ be a trajectory of (D, Γ). Then, there exists an answer set M of $\Pi^k(D, \Gamma)$ such that $s_i = s(\alpha_i, M)$, $i > 0$.*

Proof. See Appendix A.

It is easy to see that each answer set of the program $\Pi(D, \Gamma)$ corresponds to an evolution tree whose paths are possible trajectories of the domain specified by the action theory (D, Γ). The multiplicity of answer sets is due to the fact that an action theory with static causal laws may be non-deterministic. The size of the evolution tree encoded is exponential in the maximal length of the trajectories encoded in the tree (for a set of k actions and for a maximal trajectory length m, the size is $O(k^m)$). In practice, we are typically interested in generating only one of such branches that meet some desired requirements (e.g., satisfy a given set of preferences). When the action theory is deterministic, there is only one possible evolution tree of the domain. As such, we have the following corollary.

Corollary 1 *For a consistent, complete, and* deterministic *action theory (D, Γ)—i.e., $\Phi(a, s)$ has at most one element for each action a and state s—the program $\Pi(D, \Gamma)$ has a unique answer set.*

5 Planning with Preferences using $\Pi(D, \Gamma)$

It follows from Theorems 3 and 4 that trajectories achieving a formula Δ can be computed using $\Pi(D, \Gamma)$. Given an answer set M of $\Pi(D, \Gamma)$, if $s(\alpha, M)$ satisfies Δ, then α is a trajectory achieving Δ. Since an answer set of $\Pi(D, \Gamma)$ can contain other trajectories that do not achieve Δ, we propose to introduce additional rules in $\Pi(D, \Gamma)$, with the purpose of extracting the trajectories achieving Δ from an answer set. For simplicity[6], let us assume that Δ is a conjunction of fluent literals, i.e., $\Delta = f_1 \wedge \ldots \wedge f_n$. In this case, the set of rules

$$rule(goal, goal(\beta), [f_1(\beta), \ldots, f_n(\beta)]), \tag{22}$$

is added to $\Pi(D, \Gamma)$, where β is a sequence of actions. We call the new program $\Pi(D, \Gamma, \Delta)$. For an answer set M of $\Pi(D, \Gamma, \Delta)$ and a sequence of actions $\beta = a_1, \ldots, a_k$, let

$$tr(\beta, M) = s_0 a_1 s(\beta_1, M) a_2 \ldots s(\beta_{k-1}, M) a_k s(\beta, M)$$

where β_i is the prefix of length i of β. The next corollary follows immediately from Theorems 3 and 4.

Corollary 2 *Let (D, Γ) be a consistent and complete action theory, Δ be a conjunction of fluent literals, and M be an answer set of $\Pi(D, \Gamma, \Delta)$. Then, for every action sequence β, if $goal(\beta) \in M$, then $tr(\beta, M)$ is a trajectory achieving Δ.*

The significance of this corollary is that it allows us to single out trajectories for Δ from other trajectories in an answer set. Each trajectory for Δ is a possible plan to achieve it. In many situations, it is desirable to find one, among several possible trajectories, that satisfies certain constraints. For example, *ride a bus* and *take a 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 refer to these user-defined biases towards certain trajectories as *preferences between trajectories*. We will show next how $\Pi(D, \Gamma, \Delta)$ can be modified to deal with two different types of preferences between trajectories—i.e., preferences between actions and preferences between final states. In the process, we extend the prioritized default theory for representing the preferences between rules.

5.1 Preferences Between Rules

Whenever we express the fact that we do not prefer a rule r, we mean that we do not want to use r. This does not necessarily mean that r cannot be applied, but it simply means that if r can be replaced, then we prefer to do so. For this reason, we use literals of the form

$$block(r, [l_1, \ldots, l_m]) \tag{23}$$

[6]Similar encoding can be provided for an arbitrary fluent formula.

in the prioritized default theory 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,

$$block(r_1, body(r_2))$$

can be used to express the fact that we prefer to use r_2 instead of r_1—i.e., if the body of the rule r_2 is satisfied, we block the use of the rule r_1.

To implement the new type of rules in prioritized default theory, we replace rule (10) with the following rule:

$$holds(L) \quad \leftarrow \quad rule(R, L, Body), hold(Body), not\ blocked(R). \tag{24}$$

and add the next rule to the set of independent rules $\mathcal{P}$ (recall that $\mathcal{P}$ consists of the rules (10)-(15)):

$$blocked(R) \quad \leftarrow \quad block(R, Body), hold(Body). \tag{25}$$

This is used to block the application of the rule R. By $\mathcal{P}^b$ we denote the set of rules (10)-(15) and (24)-(25). 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. Next, we show how this extension to prioritized default theories can be used to express preferences between actions and preferences between final states.

For a prioritized default theory T, let $bl(T)$ denote the set of literals of the form $block(.)$ in T. The next theorem shows that if there is no preference between rules, the new set of rules behaves exactly as the old one.

Theorem 5 *For a prioritized default theory T with $bl(T) = \emptyset$, $T \cup \mathcal{P}$ and $T \cup \mathcal{P}^b$ are equivalent.*

Proof. Let Q be the set of rules of type (25). It is easy to see that if $bl(T) = \emptyset$ then $T \cup \mathcal{P}^b$ is equivalent to $T \cup \mathcal{P}^b \setminus Q$. Splitting the program (see Appendix B) with the set of literals X of the form $blocked(.)$ yields the empty program as the top and $T \cup \mathcal{P}$ as the evaluation of $T \cup \mathcal{P}^b \setminus Q$ with respect to $\langle \emptyset, X \rangle$. The splitting theorem implies that each answer set of $T \cup \mathcal{P}^b \setminus Q$ (and hence, of $T \cup \mathcal{P}^b$) is an answer set of $T \cup \mathcal{P}$ and vice versa. $\Box$.

The next lemma, useful in proving properties of programs with preferences between actions and formulae, relates the applicability of a rule in an answer set with the set of preferences over the rules.

Lemma 1 *Let T be a prioritized default theory. Let $rule(r, l, body)$ be a rule in T such that l does not occur in the head of any other rule or default in T. Then, for every answer set M of $T \cup \mathcal{P}^b$, if $blocked(r) \in M$ then $holds(l) \not\in M$.*

Proof. Since M is an answer set of $T \cup \mathcal{P}^b$ and $blocked(r) \in M$, there exists no rule in $T \cup \mathcal{P}^b$ whose head is $holds(l)$ and whose body is satisfied by M. This implies that $holds(l) \not\in M$. $\Box$

5.2 Preferences between Actions

As we have discussed earlier, an agent might prefer an action over some others for several reasons. 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 $\alpha = s_0 a_1 s_1 \ldots, a_n s_n$ is said to be* preferred *to a trajectory $\beta = s_0 b_1 s'_1 \ldots, b_m s'_m$ with respect to a set of action preferences $Pref$, denoted by $\alpha \prec_{Pref} \beta$, if*

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

Definition 2 (Most Preferred Trajectory) *A trajectory $\alpha = s_0 a_1 s_1 \ldots a_n s_n$ is said to be a* most preferred *trajectory with respect to a set of preferences $Pref$ if there exists no trajectory β such that $\beta \prec_{Pref} \alpha$.*

Remark 1 *$\prec_{Pref}$ is an antisymmetric, transitive, and irreflexive relation.*

Example 3 *Let us revisit the example of Section 2.3. If an employee wants to leave the building he will typically prefer to approach the front entrance (to avoid triggering the alarm):*

$$Pref = \{prefer(goto(frontdoor), goto(backdoor))\}$$

Similarly, if a thief is in the building and he knows that the backdoor has an alarm, he will clearly want to avoid opening it:

$$Pref = \{prefer(X, open(backdoor)) \ : \ X \in \mathbf{A} \land X \neq open(backdoor)\}.$$

We will now show how the preferences over actions can be enforced in the extended framework of prioritized default theories. Given an action theory (D, Γ), a goal Δ, and a set of preferences over actions $Pref$, we add to $\Pi(D, \Gamma, \Delta)$ the rules that block the execution of b whenever a (an action preferred to b) can be used. More precisely,

- For each preference $prefer(a, b)$ in $Pref$, and for each pair of sequences of actions α and β such that a occurs in α and $\alpha_{|\beta|} \circ a \sqsubseteq \alpha$: the set of rules of the form

  $$block(executable(b, \beta), [goal(\alpha)]) \tag{26}$$

  belongs to $\Pi(D, \Gamma, \Delta)$.

The next theorem shows that adding (26) to $\Pi(D, \Gamma, \Delta)$ will eliminate non-preferred trajectories from its conclusions.

Theorem 6 *Let (D,Γ) be a consistent and complete action theory, Δ be a conjunction of fluent literals, and $Pref$ be a set of action preferences. For every action sequence γ, if $\Pi(D,\Gamma,\Delta) \models goal(\gamma)$, then for every answer set M of $\Pi(D,\Gamma,\Delta)$ we have that $tr(\gamma, M)$ is a most preferred trajectory achieving Δ.*

Proof. $\Pi(D,\Gamma,\Delta) \models goal(\gamma)$ implies that $goal(\gamma)$ belongs to every answer set of $\Pi(D,\Gamma,\Delta)$. Since M is an answer set of $\Pi(D,\Gamma,\Delta)$, by Corollary 2 we have that $\gamma' = tr(\gamma, M)$ is a trajectory achieving Δ. It remains to be shown that γ' is indeed a most preferred trajectory. Let us assume that there exists a trajectory α such that $\alpha \prec_{Pref} \gamma'$, i.e., there exists an action a in α and and action b in γ' such that $prefer(a,b) \in Pref$.

From Theorem 3, we know that there exists an answer set M' of $\Pi(D,\Gamma,\Delta)$ such that $goal(\alpha) \in M'$. This means that M' satisfies the body of rules of the form (26). Hence, every rule or default, whose body contain $possible(b,\delta)$, where $\delta \circ b \sqsubseteq \gamma$, is not applicable in M'. It follows from Lemma 1 that $goal(\gamma) \notin M'$; this contradicts the fact that $goal(\gamma) \in M'$. □

The following corollary follows from the above theorem and Corollary 1.

Corollary 3 *Let (D,Γ) be a consistent, complete, and deterministic action theory, Δ be a conjunction of fluent literals, and M be the answer set of $\Pi(D,\Gamma,\Delta)$ with a set of preferences $Pref$. For every action sequence β, if $goal(\gamma) \in M$ then $tr(\gamma, M)$ is a most preferred trajectory achieving Δ.*

Example 4 *Consider an action theory (D,Γ), where D consists of the propositions*

$join$	**causes**	f	**if**	g, h
b	**causes**	g	**if**	$true$
c	**causes**	h	**if**	$true$
dir	**causes**	f	**if**	$\neg g$
$join$	**executable_if**	$true$		
b	**executable_if**	$true$		
c	**executable_if**	$true$		
dir	**executable_if**	$true$		
$\neg g$	**by_default**			

while the initial state Γ contains:

initially $\neg g$
initially $\neg f$
initially $\neg h$

Let us assume that the goal we desire to achieve is f. Different trajectories lead to the desired goal, e.g.,

$$\alpha = \overbrace{\{\neg f, \neg g, \neg h\}}^{s_0}\ c\ \overbrace{\{\neg f, h, \neg g\}}^{s_1}\ b\ \overbrace{\{\neg f, h, g\}}^{s_2}\ join\ \overbrace{\{f, h, \neg g\}}^{s_3}$$

$$\beta = \underbrace{\{\neg f, \neg g, \neg h\}}_{s_0}\ dir\ \underbrace{\{f, \neg g, \neg h\}}_{s_4}$$

Let us introduce the preferences

$$Pref = \{prefer(dir, b), prefer(dir, c)\}$$

which leads to

$$\beta = s_0\, dir\, s_4 \prec_{Pref} s_0\, c\, s_1\, b\, s_2\, join\, s_3 = \alpha.$$

This preference will lead to the introduction of the facts

$$\begin{array}{c} block(executable(b, \beta), [f(\alpha)]) \\ block(executable(c, \beta), [f(\alpha)]) \end{array}$$

for each sequence of actions β and for each α such that $\alpha_{|\beta|} \circ dir \sqsubseteq \alpha$. For example, the facts

$$\begin{array}{c} block(executable(b, []), [f(dir)]) \\ block(executable(c, []), [f(dir)]) \end{array}$$

will prevent the execution of b and c when the goal is reachable by dir from the initial situation.

The next example presents a situation in which a preferred trajectory is not a conclusion of the program $\Pi(D, \Gamma, \Delta)$.

Example 5 *Let (D, Γ) be an action theory, where the set of fluents contains $\{g, f, h, l\}$ and the set of actions is equal to $\{a, b, c\}$. The action theory D contains the dynamic causal law and executability condition:*

$$\begin{array}{llllll} a & \textbf{causes} & f & & \textbf{if} & \neg f \\ b & \textbf{causes} & l & & \textbf{if} & f, g \\ c & \textbf{causes} & l & & \textbf{if} & f, h \\ a & \textbf{executable\_if} & \neg f, \neg g, \neg h & & & \\ b & \textbf{executable\_if} & f, g & & & \\ c & \textbf{executable\_if} & f, h & & & \end{array}$$

and the static causal laws:

$$\begin{array}{c} g \textbf{ if } f, \neg h \\ h \textbf{ if } f, \neg g \end{array}$$

The initial state Γ is defined by the propositions:

$$\begin{array}{l} \textbf{initially } \neg f \\ \textbf{initially } \neg g \\ \textbf{initially } \neg h \\ \textbf{initially } \neg l \end{array}$$

We would like to have $\Delta = l$. It is easy to see that

$$\alpha = \underbrace{\{\neg f, \neg g, \neg h, \neg l\}}_{s_0}\ a\ \underbrace{\{f, g, \neg h, \neg l\}}_{s_1}\ b\ \underbrace{\{f, g, \neg h, l\}}_{s_2}$$

and

$$\beta = \underbrace{\{\neg f, \neg g, \neg h, \neg l\}}_{s_0}\ a\ \underbrace{\{f, h, \neg g, \neg l\}}_{s_1'}\ c\ \underbrace{\{f, h, \neg g, l\}}_{s_2'}$$

are two possible trajectories that achieve the goal Δ. Assume that we prefer b to c, i.e., $Pref = \{prefer(b, c)\}$. Obviously, $\alpha \prec_{Pref} \beta$.

It is easy to see that for every answer set of $\Pi(D, \Gamma, \Delta)$, if it contains $goal(\gamma)$ then a must be the first action of γ. Thus, there exists one answer set of $\Pi(D, \Gamma, \Delta)$, say M_0, in which $s_1' = s(a, M_0)$. Obviously, the only action that is executable in this state is c and therefore $goal(\beta) \in M_0$ and $goal(\alpha) \notin M_0$, i.e., $\Pi(D, \Gamma, \Delta) \not\models goal(\alpha)$.

Observe that the encoding of action preferences in prioritized default theory offers scope for a number of generalizations. For example, the same scheme allows us to encode *conditional action preferences*, where the preference between actions is considered only if a given set of conditions is satisfied. We can extend the syntax of action preferences as follows:

$$prefer(a, b) \textbf{ if } q_1, \ldots, q_m$$

where $q_1, \ldots, q_m$ is a conjunction of fluent literals. The encoding in prioritized default theory is straightforward: for each

$$prefer(a, b) \textbf{ if } q_1, \ldots, q_m$$

we add the set of block rules of the form

$$block(executable(b, \beta), [goal(\alpha), q_1(\beta), \ldots, q_m(\beta)])$$

where a occurs in α and $\alpha_{|\beta|} \circ a \sqsubseteq \alpha$.

5.3 Preferences between Formulae

The second type of preferences that we consider consists of preferences between formulae. Unlike preferences between actions, 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 the agent that is trying to exit the building, described in Section 2.3. He might prefer to exit through the front door since this will avoid alerting the police ($\neg police\_alerted$). Here, the primary goal of the agent is to leave the building ($\neg\ inside$), and his soft constraint is to avoid alerting the police. The trivial choice would be to go to the front door. Going to the back door should be used as the last resource. This type of preference can be added to an action theory by introducing preferences of the form

$$\varphi_1 \prec \varphi_2. \tag{27}$$

where φ_1 and φ_2 are fluent formulae. Given a set of fluent formulae $Pref$ expressing final state preferences, we assume that $(Pref, \prec)$ is a total order—i.e., the set of preference formulae can be written as

$$\varphi_1 \prec \varphi_2 \prec \cdots \prec \varphi_k$$

This condition is acceptable in a variety of situations—indeed, many proposals in the literature dealing with preferences in logic programming, e.g., [3, 5], extend partial orders to total orders, and use such total orders in computing the preferred answers. We will explore in future works how to deal directly with partial orders. In addition, we require that for each $1 \leq i \leq k-1$ the following holds:

$$\varphi_i \Rightarrow \neg\varphi_{i+1} \wedge \ldots \wedge \neg\varphi_k.$$

The Definitions 1 and 2—preferred trajectories and most preferred trajectories—can be extended to the case of preferences between formulae as follows.

Definition 3 (Final State Preferred Trajectory) *A trajectory $\alpha = s_0 a_1 s_1 \ldots, a_n s_n$ is said to be* preferred *to a trajectory $\beta = s_0 b_1 s'_1 \ldots, b_m s'_m$ with respect to a set of preferences between formulae $Pref$ if*

1. *there exists $\varphi_1 \prec \varphi_2 \in Pref$ such that $s_n \models \varphi_1$ and $s'_m \models \varphi_2$ (denoted by $\alpha \prec_{\varphi_1 \prec \varphi_2} \beta$), and*
2. *for every $\varphi_1 \prec \varphi_2 \in Pref$, $\beta \not\prec_{\varphi_1 \prec \varphi_2} \alpha$.*

We will denote the fact that α is preferred to β w.r.t. $Pref$ with the notation $\alpha \prec_{Pref} \beta$.

Definition 4 (Final State Most Preferred Trajectory) *A trajectory $\alpha = s_0 a_1 s_1 \ldots a_n s_n$ is said to be a* most preferred *trajectory with respect to a set of preferences between formulae $Pref$ if there exists no trajectory β such that $\beta \prec_{Pref} \alpha$.*

Remark 2 *The relation $\prec_{Pref}$ between trajectories is a partial order.*

To encode this type of preferences, we add to $\Pi(D, \Gamma)$ a set of rules to block the execution of an action leading to φ_2 as long as φ_1 is satisfied by other trajectories. We assume that, for each action a, for each goal formula Δ, and for each formula φ_2, we can compute a formula ϕ_{a,φ_2} such that if a is executed in a state satisfying ϕ_{a,φ_2} then $\Delta \wedge \varphi_2$ holds in the successor state. There are well-known regression techniques that have been proposed in the literature to determine such formulae (see e.g. [30]). In that case, we will block the execution of a if there exists another trajectory satisfying φ_1. For each preference $\varphi_1 \prec \varphi_2$ in P, we will add to $\Pi(D, \Gamma, \Delta)$ the following set of rules

$$block(executable(a, \gamma), [goal(\beta), \varphi_1(\beta), \phi_{a,\varphi_2}(\gamma)]) \tag{28}$$

where γ and β are sequences of actions. Additionally, we add to $\Pi(D, \Gamma, \Delta)$ rules for reasoning about ϕ_{a,φ_2} which are similar to (22). With a slight abuse of notation, we use $\Pi(D, \Gamma, \Delta)$ to denote the program with rules of the form (26) and (28) when formulae preferences are present.

Example 6 *Consider the action theory (D, Γ) where D contains the propositions*

a	**causes**	f	**if**	g
b	**causes**	g	**if**	$\neg g$
c	**causes**	f	**if**	$\neg h$
a	**executable_if**	$true$		
b	**executable_if**	$true$		
c	**executable_if**	$true$		
h	**if**	g		

and Γ contains

$$\textbf{initially } \neg f$$
$$\textbf{initially } \neg g$$
$$\textbf{initially } \neg h$$

The goal is f and we are assuming the presence of a preference

$$Pref = \{\neg h \prec h\}.$$

It is easy to see that $\alpha = s_0 c s_1$ where $s_1 = \{f, \neg h, \neg g\}$ is a most preferred trajectory. $\beta = s_0 b s'_1 a s_2$ is another trajectory achieving f but we have that $\alpha \prec_{Pref} \beta$.
One of the facts generated to handle this preference is

$$block(executable(a, \gamma), [f(\beta), \neg h(\beta), \phi_{a,h}(\gamma)])$$

for sequences of actions γ, β. For example, for

$$\gamma = b$$
$$\beta = c$$

we will obtain

$$block(executable(a, b), [f(c), \neg h(c), g(b)])$$

since executing action a in a state satisfying g will lead to a state that satisfies both the goal (f) as well as the right-hand side of the preference (h). This prevents the execution of a after b, i.e., this will eliminate the trajectory β from answer sets of $\Pi(D, \Gamma, \Delta)$.
It is instructive to see that the action sequence c, i.e., the trajectory α is not removed from any answer sets of $\Pi(D, \Gamma, \Delta)$. Assume the contrary, i.e., we have an answer set M with $goal(c) \not\in M$. This can happen only if the rule of the form (28) (for the action c)

$$block(executable(c, []), [f(\delta), \neg h(\delta), \phi_{c,h}([])])$$

has its body satisfied by M. Since the execution of c in any state will not change the value of h, we have that $\phi_{c,h} = h$. This implies that $h([]) \in M$, i.e., h is true in the initial state — a contradiction, which implies that $goal(c)$ belongs to every answer set of $\Pi(D, \Gamma, \Delta)$.

The following theorem is similar to Theorem 6.

Theorem 7 *Let (D, Γ) be a consistent and complete action theory, Δ be a conjunction of fluent literals, and $Pref$ be a set of preference between formulae. For each non-empty action sequence β, if $\Pi(D, \Gamma, \Delta) \models goal(\beta)$ then for every answer set M of $\Pi(D, \Gamma, \Delta)$ we have that $tr(\beta, M)$ is a most preferred trajectory achieving Δ.*

Proof. $\Pi(D, \beta, \Delta) \models goal(\beta)$ implies that $goal(\beta)$ belongs to every answer set of $\Pi(D, \beta, \Delta)$. Since M is an answer set of $\Pi(D, \Gamma, \Delta)$, by Corollary 2 we have that $tr(\beta, M)$ is a trajectory achieving Δ. It remains to be shown that $\beta' = tr(\beta, M)$ is indeed a most preferred trajectory. Assume the contrary, $\alpha \prec_{\varphi_1 \prec \varphi_2} \beta$ for some preference $\varphi_1 \prec \varphi_2$. From Theorem 3, we know that there exists an answer set M' of $\Pi(D, \Gamma, \Delta)$ such that $goal(\alpha) \in M'$. Let $\alpha' = tr(\alpha, M)$.

Because $\alpha' \prec_{\varphi_1 < \varphi_2} \beta'$, we have that $\varphi_1(\alpha) \in M'$. Since β is a non-empty action sequence, there exists an action a and an action sequence γ such that $\beta = \gamma \circ a$. It follows from the construction of M', $\phi_{a,\varphi_2}(\gamma) \in M'$. This means that the body of the rule (28) is satisfied by M'. Thus, $holds(possible(a, \gamma)) \notin M'$ (Lemma 1); this contradicts the fact that $goal(\beta) \in M'$. $\Box$

The above theorem shows that if $goal(\beta)$ is contained in every answer set of $\Pi(D, \Gamma, \Delta)$ then β is a most preferred trajectory with respect to $Pref$. Due to the presence of distinct answer sets, it is possible to find a trajectory γ such that $goal(\gamma)$ belongs to some but not all answer sets of $\Pi(D, \Gamma, \Delta)$. In this case, we cannot guarantee that γ is a most preferred trajectory. This is shown in the next example.

Example 7 *Consider again the planning problem (D, Γ, Δ) in Example 5. Recall that the two trajectories*

$$\alpha = \underbrace{\{\neg f, \neg g, \neg h, \neg l\}}_{s_0} \ a \ \underbrace{\{f, g, \neg h, \neg l\}}_{s_1} \ b \ \underbrace{\{f, g, \neg h, l\}}_{s_2}$$

and

$$\beta = \underbrace{\{\neg f, \neg g, \neg h, \neg l\}}_{s_0} \ a \ \underbrace{\{f, h, \neg g, \neg l\}}_{s'_1} \ c \ \underbrace{\{f, h, \neg g, l\}}_{s'_2}$$

both achieve the goal l. Consider the set of preferences $Pref = \{\neg h \prec \neg g \wedge h\}$. Again, we have that $\alpha \prec_{Pref} \beta$. Similar argument as in Example 5 shows that $\Pi(D, \Gamma, \Delta)$ has an answer set which contains $goal(\beta)$ and does not contain $goal(\alpha)$.

The following corollary is similar to Corollary 3.

Corollary 4 *Let (D, Γ) be a consistent, complete, and deterministic action theory, Δ be a conjunction of fluent literals, and M be an answer set of $\Pi(D, \Gamma, \Delta)$ with a set of preferences between formulae $Pref$ where $\prec$ is a total order. For every action sequence β, if $goal(\gamma) \in M$ then $tr(\gamma, M)$ is a most preferred trajectory achieving Δ.*

Example 8 *Let us consider the example of section 2.3. The person trying to exit the building may prefer to use the key for the backdoor than asking the janitor for another key:*

$$haveKey(backdoor) \prec haveKey(frontdoor)$$

As another example, it may be likely that the person wants to leave the building without alerting the police

$$\neg police\_alerted \prec police\_alerted$$

Similarly to what discussed in the case of action preferences, we can extend the notation used for formulae preferences to encode conditional preferences. The notation we suggest is:

$$\varphi_1 \prec \varphi_2 \ \textbf{if} \ p_1, \ldots, p_n$$

where φ_1, φ_2 are preferences between fluent formulae and $p_1, \ldots, p_n$ are fluent literals. Intuitively, the preference states that we are interested in considering this particular preference only if the final state satisfies the literals $p_1, \ldots, p_n$. The modification to the encoding in prioritized default theory is the following:

$$block(executable(a, \gamma), [goal(\beta), \varphi_1(\beta), \phi_{a,\varphi_2}(\gamma), p_1(\beta), \ldots, p_n(\beta)]).$$

6 Computing the Entailment Relation $\models$ Using SMODELS

The program $\Pi^k(D,\Gamma)$ can be implemented using a solver for answer set programming; in particular, in this work we experimented our ideas using the SMODELS [25] system. 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. An automated translator to convert $\Pi^k(D,\Gamma)$ to a SMODELS program has been devised and can be found at `www.cs.nmsu.edu/lldap/Preferences`. Below, we describe the translation and prove its correctness. Appendix C presents an example of a translation using the scheme described in this section.

6.1 Encoding $\Pi^k(D,\Gamma)$ as a SMODELS Program

In this section, we will present the encoding of $\Pi^k(D,\Gamma)$ as a SMODELS program, denoted by $SM^k(D,\Gamma)$. In the next section, we will show how $SM^k(D,\Gamma)$ can be used in finding trajectories and preferred trajectories using SMODELS. Since $\Pi^k(D,\Gamma)$ consists of two parts, the set of rules $R^k = \bigcup_{|\alpha|\le k} R_\alpha$ and the set of rules (10)-(15), we divide the translation in two parts. The first part deals with the default theory (R^k) while the second part deals with the rules (10)-(15) ($\Pi^k(D,\Gamma)\setminus R^k$).

6.1.1 SMODELS Encoding For R^k

Observe that we can divide literals in the language underlying R^k, denoted by $\mathcal{L}$, into three types:

(A.i) $f(\gamma)$ where f is a fluent and γ is a sequence of actions;

(A.ii) $possible(a,\gamma)$ where a is an action and γ is a sequence of actions;

(A.iii) $dynamic(f,a,\gamma)$, $causal(f,a,\gamma)$, $executable(a,\gamma)$, $def(f,a,\gamma)$, and $inertial(f,a,\gamma)$ – which are names for rules and defaults in the prioritized default theories.

Similarly, every list occurring in $\Pi(D,\Gamma)$ contains only literals of the types (A.i)-(A.ii) and is of the following form:

(L.i) $[p_1(\gamma),\dots,p_m(\gamma)]$ where p_i's are fluent literals and γ is a sequence of actions;

(L.ii) $[p_1(\gamma),\dots,p_m(\gamma),possible(a,\gamma)]$ where p_i's are fluent literals, a is an action, and γ is a sequence of actions; or

(L.iii) $[p_1(\gamma\circ a),\dots,p_m(\gamma\circ a),possible(a,\gamma)]$ where p_i's are fluent literals, a is an action, and γ is a sequence of actions.

It is worth noticing that each list in R^k can be divided into two parts. The first part consists of literals of the form (A.i) and the second part is either empty (for (L.i)) or consists of a single element of the form (A.ii). Furthermore, if $x = [l_1,\dots,l_m]$ is a list of literals of the form (A.i) obtained through this splitting then l_i's have the same sequence of actions as their last parameter. The language underlying the prioritized default theory of the program $SM^k(D,\Gamma)$, denoted by $\mathcal{L}^{SM}$, is defined as follows.

- For each $l \in \mathcal{L}$ of the types (A.i)-(A.iii), $\mathcal{L}^{SM}$ contains l^+, which is obtained from l by replacing γ with $|\gamma|$; for instance, $f(a \circ b)$ becomes $f(2)$, $possible(a, [])$ becomes $possible(a, 0)$, etc.
- For each list y in $\mathcal{L}$, $\mathcal{L}^{SM}$ contains a new and distinguished atom $n_{|y|}(m)$ where (i) $|y|$ is the list of fluent literals occurring in y; and (ii) m is the length of the sequence of actions that appears as the last parameter in literals of the form (A.i) belonging to y; for example:
  - the list $[f(a), g(a)]$ is associated with the atom $n_{[f,g]}(1)$, where $n_{[f,g]}$ is a new and distinguished predicate name that does not appear in the language $\mathcal{L}$,
  - the list $[f(a \circ c), \neg g(a \circ c), possible(b, a \circ c)]$ is associated with the atom $n_{[f,\neg g]}(2)$, and
  - the list $[f(a \circ c), g(a \circ c), possible(c, a)]$ is associated with the atom $n_{[f,g]}(2)$.

Intuitively, the integer associated to each literal of $\mathcal{L}^{SM}$ denotes a time stamp on a linear time line. For example, $f(2)$ represents the fact that f is true at the time moment 2. $possible(a, 3)$ says that it is possible to execute the action a at the time moment 3. As such, while R^k that represents a tree of possible trajectories whose length is at most k, R^k_{sm} — the SMODELS encoding of R^k — only represents a possible trajectory. Given a trajectory $s_0 a_1 s_1 \dots a_n s_n$, the definition of a trajectory says that a_j must be executable in s_{j-1}. For this reason, we will drop literals of the form (A.ii) from the lists occurring in rules of R^k and give them a special treatment. We will introduce literals of the from $occ(a, i)$ where a is an action and i is an integer to indicate that a occurs at the time moment i and introduce a constraint stating that an action can occur only if it is executable. The translation is done as follows.

- For each atom of the form (16), the following rule belongs to R^k_{sm}:

$$rule(dynamic(f, a, m), f(m+1), n(m)) \leftarrow occ(a, m) \tag{29}$$

  where $m = |\beta|$ and $n(m)$ is the atom associated to the list of atoms of the from (A.i) occurring in (16).

- For each atom of the form (17), the following atom belongs to R^k_{sm}:

$$rule(executable(a, m), possible(a, m), n(m)) \tag{30}$$

  where $m = |\beta|$ and $n(m)$ is the atom associated to the list occurring in (17).

- For each atom of the form (18), the following atom belongs to R^k_{sm}:

$$rule(causal(f, a, m), f(m+1), n(m+1)) \leftarrow occ(a, m) \tag{31}$$

  where $m = |\beta|$ and $n(m)$ is the atom associated to the list of atoms of the from (A.i) occurring in (18).

- For each atom of the form (19), the following atom belongs to R^k_{sm}:

$$default(def(f, a, m), f(m+1), true) \leftarrow occ(a, m) \tag{32}$$

  where $m = |\beta|$ and $n(m)$ is the atom associated to the list of atoms of the from (A.i) occurring in (19).

- For each atom of the form (20), the following atom belongs to R^k_{sm}:

$$default(inertial(f,a,m), f(m+1), n(m)) \leftarrow occ(a,m) \tag{33}$$

  where $m = |\beta|$ and $n(m)$ is the atom associated to the list of atoms of the from (A.i) occurring in (20).

- For each atom of the form (21), the atom

$$holds(f(0)) \tag{34}$$

  belongs to R^k_{sm}.

Finally, for every action a, we add the constraint

$$\leftarrow occ(a,T), not\ holds(possible(a,T)) \tag{35}$$

to R^k_{sm} and for each atom $n_{[l_1,\ldots,l_n]}$ representing the list $[l_1,\ldots,l_n]$, we add rules of the following form

$$hold(n_{[l_1,\ldots,l_n]}(T)) \leftarrow holds(l_1(T)),\ldots,holds(l_n(T)). \tag{36}$$

to R^k_{sm}, where T is the time variable.

Notice the difference between R^k and R^k_{sm}. For an action theory (D,Γ), R^k_{sm} represents a possible history while R^k represents an evolution tree. As such, the size of R^k_{sm} is rather small comparing to that of R^k.

Example 9 *The dynamic causal law*

$$opendoor(backdoor)\ \textbf{causes}\ open(backdoor)\ \textbf{if}\ at(backdoor)$$

is encoded as the logic programming rule:

$$\begin{aligned} rule(dynamic(open(backdoor), opendoor(backdoor), T), open(backdoor, T+1), n_{[at(backdoor)]}(T)) \\ \leftarrow occ(opendoor(backdoor), T), time(T). \end{aligned}$$

for values of T which are legal plan lengths (the predicate $time$ is used to limit T to acceptable values). In addition, the rule for $hold$ will be specialized:

$$hold(n_{[at(backdoor)]}(T)) \leftarrow time(T), holds(at(backdoor,T))$$

and the constraint

$$\leftarrow occ(opendoor(backdoor),T), not\ holds(possible(opendoor(backdoor),T))$$

is added to the SMODELS *encoding.*

6.1.2 SMODELS **Encoding For** $\Pi^k(D,\Gamma) \setminus R^k$

Since SMODELS does not allow the list operator and the rule (36) effectively replaces the two rules (12)-(13), the only thing we need to do in encoding $\Pi^k(D,\Gamma) \setminus R^k$ is to remove the two rules (12)-(13) from $\mathcal{P}$. Other rules do not change. Thus, the program $SM^k(D,\Gamma)$ consists of R^k_{sm} and the rules (10)-(11) and (14)-(15).

6.1.3 Property of $SM^k(D,\Gamma)$

We now discuss a property of $SM^k(D,\Gamma)$. As we have pointed out earlier, in encoding $\Pi^k(D,\Gamma)$, we replace a sequence of actions with its length. As such each answer set of $SM^k(D,\Gamma)$ corresponds to only one trajectory of the action theory while each answer set of Π^k corresponds to a possible evolution tree. We will now show that this difference will be eliminated when we fix the trajectory in both programs. Let (D,Γ) be a complete and consistent action theory, and let $\alpha = a_1,\ldots,a_k$ be a sequence of actions. Let $P^\alpha(D,\Gamma)$ be the program consisting of

- the rules (10)-(15), and
- the set of rules R_α.

The next theorem relates the program $P^\alpha(D,\Gamma)$ and $SM^k(D,\Gamma)$.

Theorem 8 *Let (D,Γ) be a consistent and complete action theory and $\alpha = a_1,\ldots,a_k$ be a sequence of actions. Then,*

1. *if M is an answer set of $P^\alpha(D,\Gamma)$, then $SM^k(D,\Gamma) \cup \{occ(a_i, i-1) \mid i = 1,\ldots,k\}$ has an answer set M' with the property that*

$$M \models holds(f(a_1 \circ \cdots \circ a_i)) \text{ if and only if } M' \models holds(f(i))$$

2. *if M is an answer set of $SM^k(D,\Gamma) \cup \{occ(a_i, i-1) \mid i = 1,\ldots,k\}$ then there exists an answer set M' of $P^\alpha(D,\Gamma)$ such that*

$$M \models holds(f(i)) \text{ if and only if } M' \models holds(f(a_1 \circ \cdots \circ a_i)).$$

Proof. See Appendix A. □

6.2 Finding a Trajectory Using $SM^k(D,\Gamma)$

The discussion in the previous section shows that $SM^k(D,\Gamma)$ can be used to compute the entailment relation of (D,Γ). In this section, we discuss the use of $SM^k(D,\Gamma)$ in finding a trajectory $s_0a_1\ldots a_ks_k$ that satisfies the following properties:

1. $s_k \models \Delta$ for some given fluent formula Δ—this means that the trajectory is a possible plan to accomplish the goal Δ;
2. the trajectory $s_0a_1\ldots a_ks_k$ satisfies some soft constraints that are expressed as preferences between actions or between fluent formulae.

6.2.1 Finding A Trajectory for Δ

Let Δ be a conjunction of fluent literals $f_1 \wedge \ldots \wedge f_k$[7]. We are interested in finding a trajectory $s_0a_1\ldots a_ks_k$ for Δ. As it is customary in answer set planning, we add to $SM^k(D,\Gamma)$ the set of

[7]Fluent formula can be dealt with as in [34].

rules to generate action occurrences and to represent the goal. This set of rules consists of:

$$
\begin{array}{rcll}
 & \leftarrow & not\ goal(k). & (37)\\
goal(T) & \leftarrow & time(T), holds(f_1(T)), \ldots, holds(f_k(T)). & (38)\\
1\{occ(A,T) : action(A)\}1 & \leftarrow & time(T), T < k. & (39)
\end{array}
$$

In addition, we add the set of facts $\{action(a) \mid a \in \mathbf{A}\}$ to $SM^k(D,\Gamma)$ which specifies the domain of actions needed for the grounding of rules (39). Intuitively, rule (38) is used to express under what conditions the goal can be considered to be satisfied at time T. Rule (37) is an answer set constraint [25] used to reject answer sets that do not satisfy the goal at time n. Rule (39) makes use of an SMODELS *choice rule* to ensure that, at each time T, the answer set includes exactly one fact of the form $occ(A,T)$, where A is an action name. Let $SM^{Plan,k}(D,\Gamma,\Delta)$ be the program consisting of the rules of $SM^k(D,\Gamma)$ and the set of rules (37)-(39), in which the time variable takes values from 0 to k. The next theorem relates trajectories satisfying the goal Δ to answer sets of $SM^{Plan,k}(D,\Gamma,\Delta)$.

Theorem 9 *For a consistent and complete action theory* (D,Γ)*,*

1. *if* $s_0a_1\ldots a_ks_k$ *is a trajectory for* Δ *then* $SM^{Plan,k}(D,\Gamma,\Delta)$ *has an answer set* M *such that*

   (a) $occ(a_i, i-1) \in M$ *for every integer* i*,* $1 \le i \le k$*, and*

   (b) $s_i = \{f \mid holds(f(i)) \in M\}$*;*

2. *if* $SM^{Plan,k}(D,\Gamma,\Delta)$ *has an answer set* M *such that*

   (a) $occ(a_i, i-1) \in M$ *for every integer* i*,* $1 \le i \le k$*, and*

   (b) $s_i = \{f \mid holds(f(i)) \in M\}$

   then $s_0a_1\ldots a_ks_k$ *is a trajectory for* Δ*.*

Proof. See Appendix A. □

6.2.2 Finding a Preferred A Trajectory: Action Preferences

Let us now encode the preferences between actions as rules of $SM^{Plan,k}(D,\Gamma,\Delta)$. For simplicity, instead of translating the set of preferences between actions into literals of the form (23) we will encode it directly as SMODELS rules. For each action preference $prefer(a,b)$, we define a rule

$$block(executable(b,T), nil) \quad \leftarrow \quad goal(length) \tag{40}$$

where nil is the name assigned to the list [] which is true at every moment of time. Intuitively, this rule prevents the action b to be used in achieving the goal. Let (D,Γ) be an action theory and P be a set of preferences on actions in D. Let $SM^{Pref,k}(D,\Gamma,\Delta)$ be the program consisting of

- the program $SM^{Plan,k}(D,\Gamma,\Delta)$, and
- the set of rules (40) with the time variable ranging between 0 and k.

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 to construct the trajectory. This encoding provides the following form of soundness:

Theorem 10 *Let (D, Γ) be a consistent and complete action theory and M be an answer set of the program $SM^{Pref,k}(D, \Gamma, \Delta)$ encoding the planning problem (D, Γ, Δ) with a set of preference P. Then, $s_0 a_1 s_1 \cdots a_n s_n$ is a most preferred trajectory satisfying Δ where*

- $occ(a_i, i-1) \in M$
- $s_i = \{f \mid holds(f(i)) \in M\}$

Proof. See Appendix A. □

Notice that the rules of the form (40) do not warrant completeness, i.e., they do not guarantee that a most preferred trajectory is found, even if one exists. For instance, when two actions a and b are possible and we have the preference $prefer(a, b)$, but the action a fails to lead to the final goal, then the program may fail to produce a trajectory. At this time it is unclear whether completeness can be achieved using this encoding of prioritized default theories with action preferences.

An alternative approach to encode preferences between actions can be developed using the SMODELS construct **maximize**. The **maximize** construct allows the programmer to associate static weights (non-negative integers, w_i) to a selected set of ground atoms (a_i):

$$\textbf{maximize}[a_1 : w_1, \cdots, a_k : w_k].$$

Intuitively, this rule instructs SMODELS to find answer set in which the sum

$$\Sigma_{i=0}^{k} |a_i| * w_i$$

is maximal where $|a_i| = 1$ if a_i is in the answer set and $|a_i| = 0$ otherwise. SMODELS makes use of branch-and-bound techniques to return an answer set with maximal weight—i.e., it maximizes the sum of the weights of the atoms satisfied by the answer set.

We can ensure that the most preferred trajectory can always be found, by adding the following optimization rule to the program $SM^{Pref,n}(D, \Gamma, \Delta)$: for each $prefer(a, b)$ and for each time point t

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

6.2.3 Finding a Preferred Trajectory: Formulae Preferences

To implement (27), for a totally ordered collection of fluent formulae $\varphi_1 \prec \ldots \prec \varphi_k$, we add the optimal rule [24]

$$\mathbf{maximize}[\varphi_1 = k, \ldots, \varphi_k = 0] \tag{41}$$

to $SM^{Plan,n}(D, \Gamma)$. We are assuming that the computation of the answer sets maximizes each rule of type (41).[8] If we want to use the current version of SMODELS, then we need to additionally

[8]Observe that the current implementation of SMODELS does not guarantee this behavior—SMODELS maximizes only the last optimal rule in the program.

require that the preference relation $\prec$ is total order over the set of fluent formulae. The correct behavior of **maximize** is guaranteed in the current implementation of the *Jsmodels* system [15]. The use of the implementation of **maximize** in *Jsmodels* allows us to make use of both types of preferences concurrently within the same domain specification.

7 Discussion and Conclusions

The advantages of making use of high level languages for the description of action theories have been highlighted by many researchers (e.g., [9, 36]). Our interest in this line of research is to enrich action theories with more complex forms of reasoning—including reasoning with preferences over trajectories and handling default and exogenous actions. In this paper we presented a formalism for reasoning about actions in the context of prioritized default theory. In the process, we developed an encoding of action theories in prioritized default theories, whose semantics coincides with the entailment relation of the action theory. It is worth noticing that prioritized default theory is 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 user-defined preferences between trajectories;
- 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).

We illustrate these last two types of actions in the following simple example.

Example 10 *Consider a mail delivering robot who drives around the city to deliver mails. The robot knows that it can pass an intersection when the traffic light is green and that it needs to stop when the traffic light is red. In this domain, the action of changing the traffic light color could be viewed as a default action which changes the color of the traffic light from* green *to* yellow*, from* yellow *to* red*, and from* red *to* green*, and so on. This action is not an action that the robot can do. It is also not an action that happens arbitrarily. Rather, its behavior can be predicted given the current situation, i.e., the light will be green in the next situation if it is currently red.*

The robot also knows that the traffic light could also be changed by an ambulance in an emergency situation. This action is an example of an exogenous action. Like a default action, exogenous actions are actions that the robot can not perform but their occurrence is rather unpredictable.

Observe that exogenous actions can be added to any of the current RAC's approaches without the need of redefining their semantics. Exogenous actions can be used to explain discrepancies between the real state of the world (represented by observations) and the predicted model of the world (represented by the effects of actions); for example, in [2], exogenous actions are used in formalizing dynamic diagnoses. For default actions, certain modifications need to be done to take into consideration their occurrences. In approaches using high-level action description languages, this would mean that the definition of the transition function needs to be revised because in a domain with default actions, the real state of the world changes even when the agent does nothing.

The previous work conducted by the authors [11] and the related work conducted by other researchers (e.g., [36]) have demonstrated the advantages of making use of more specialized forms of logic, such as prioritized default theories, for commonsense reasoning, causal reasoning, and other advanced forms of reasoning. In this work we propose to lay the foundations for using prioritized default theory for reasoning about actions and planning. Our claim is that advanced forms of reasoning about actions (including preferences, exogenous actions, and default actions) can be naturally addressed in the context of prioritized default theory—some preliminary steps in this direction can be found in [11]. In this paper we lay the foundations for this research. We accomplish this by illustrating a sound and complete translation of a high-level action language (a variation of the language $\mathcal{B}$) into prioritized default theory, along with a simple extension of prioritized default theory that allows an elegant encoding of powerful types of preferences between trajectories and management of non-inertial fluents. Our extension of prioritized default theory allows preferences between rules and formulae to be expressed. We also show how these features can be ultimately translated from prioritized default theory to answer set programming, thus allowing us to use inference engines for answer set programming (e.g., SMODELS) for planning. Further extensions to handle exogenous and default actions will be considered in our future work.

The considerations provided in this work represent also a starting point towards the treatment of more general forms of preferences. In general, an agent can have several preferences on trajectories. 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 [34]),
- adding an optimization rule
$$\mathbf{maximize}[\varphi^T = 1, not\ \varphi^T = 0]$$
to the program $SM^{Plan,n}(D, \Gamma)$, that allows us to find trajectories satisfying φ before considering those not satisfying it.

A more complete treatment of these preferences is beyond the scope of this work and will be dealt with as future work.

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 into a predefined collection of software services [27]—and in the development of Web accessibility agents for visually impaired individuals [28].

Several other approaches to dealing with preferences between logic programming rules have been proposed [3, 5, 37]. 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 M. Balduccini for the comments on various drafts of this work, and the anonymous referees for their helpful comments. Research has been supported by NSF grants EIA-0130887, CCR-9875279, HRD-9906130, EIA-0220590, and EIA-9810732.

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] C. Baral, S. McIlraith, and T.C. Son. Formulating Diagnostic Problem Solving using an Action Language with Narratives and Sensing. *Proceedings of the Knowledge Representation and Reasoning Conference*, 2000.

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

[4] S. Citrigno, T. Eiter, W. Faber, G. Gottlob, C. Koch, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The dlv system: Model generator and application frontends. In F. Bry, B. Freitag, and Seipel D., editors, *Proceedings of the 12th Workshop on Logic Programming WLP*, pages 128–137, Sep 1997.

[5] 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.

[6] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. Answer set planning under action costs. In S. Flesca and S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the Eighth European Conference on Logics in Artificial Intelligence, JELIA'02*, pages 186–197. Springer Verlag, LNAI 2424, 2002.

[7] 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.

[8] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and Peter Szeredi, editors, *Logic Programming: Proceedings of the Seventh International Conf.*, pages 579–597, 1990.

[9] M. Gelfond and V. Lifschitz. Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17:301–322, 1993.

[10] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.

[11] M. Gelfond and T.C. Son. Prioritized default theory. In *Selected Papers from the Workshop on Logic Programming and Knowledge Representation 1997*, pages 164–223. Springer Verlag, LNAI 1471, 1998.

[12] M. Ginsberg and D. Smith. Reasoning about actions I: a possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[13] P. Haddawy. A logic of time, chance, and action for representing plans. *Artificial Intelligence*, 80(1-2):243–308, 1996.

[14] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[15] H. Le, E. Pontelli, T.C. Son. An Java Based Solver for Answer Set Programming, `www.cs.nmsu.edu/~hle`, 2003.

[16] V. Lifschitz. The logic of common sense. *ACM Computing Surveys*, 27:343–345, 1995.

[17] V. Lifschitz. Answer set planning. In *International Conference on Logic Programming*, pages 23–37, 1999.

[18] 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.

[19] V. Lifschitz and H. Turner. Representing transition systems by logic programs. In *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 92–106, 1999.

[20] N. McCain and M. Turner. Causal theories of action and change. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 460–467. AAAI Press, 1997.

[21] J. McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1038–1044. Morgan Kaufmann Publishers, San Mateo, CA, 1977.

[22] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[23] K. Myers. Generating qualitatively different plans through metatheoretic biases. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. AAAI Press, 1999.

[24] 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.

[25] I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings ICLP & LPNMR*, pages 420–429, 1997.

[26] E. Pednault. ADL and the state-transition model of actions. *Journal of Logic and Computation*, 4(5):467–513, October 1994.

[27] E. Pontelli, G. Gupta, D. Ranjan, and B. Milligan. A Domain Specific Language for Solving Philogenetic Inference Problems. Technical Report TR-CS-001/2002, New Mexico State University, 2002.

[28] E. Pontelli and T. Son. Navigating HTML Tables: Planning, Reasoning, and Agents. In *Int. Conference on Assistive Technologies*. ACM Press, 2002.

[29] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

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

[31] M. Shanahan. *Solving the frame problem: A mathematical investigation of the commonsense law of inertia*. MIT press, 1997.

[32] D.E. Smith and D. Weld. Conformant GraphPlan. In *AAAI*, AAAI/MIT Press, pp. 889-896, 1998.

[33] T.C. Son and C. Baral. Formalizing Sensing Actions—A Transition Function Based Approach. In *Artificial Intelligence*, 125(1–2):19–91, 2001.

[34] T.C. Son, C. Baral, and S. McIlraith. Domain dependent knowledge in planning - an answer set planning approach. In *Proceedings of the 6th International Conference on Logic Programming and NonMonotonic Reasoning*, pages 226–239, Vienna, 2001.

[35] T.C. Son and E. Pontelli. Reasoning about actions in prioritized default theory. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the Eighth European Conference on Logics in Artificial Intelligence, JELIA'02*, pages 369–381. Springer Verlag, LNAI 2424, 2002.

[36] H. Turner. Representing actions in logic programs and default theories. *Journal of Logic Programming*, 31(1-3):245–298, May 1997.

[37] Y. Zhang and N. Foo. Answer sets for prioritized logic programs. In *Proceedings of ILPS 97*, pages 69–84, 1997.

Appendix A – Proofs

We apply the Splitting Theorem and Splitting Sequence Theorem [18] several times in our proof. To make the presentation more self-contained, the splitting theorems are included in Appendix B. Let r be a rule

$$a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n. \tag{42}$$

By $head(r)$, $body(r)$, and $lit(r)$ we denote a_0, $\{a_1, \ldots, a_n\}$, and $\{a_0, a_1, \ldots, a_n\}$, respectively. $pos(r)$ and $neg(r)$ denote the set $\{a_1, \ldots, a_m\}$ and $\{a_{m+1}, \ldots, a_n\}$, respectively. For a program π, by $lit(\pi)$ we denote the set of literals of the program π. The following lemma is also useful.

Lemma 2 *Let M be an answer set of $\Pi^k(D, \Gamma)$ and $\alpha = a_1, \ldots, a_l$, $l \leq k$, be an action sequence. If for every $j \leq l$, a_j is executable in $s(\alpha_{j-1}, M)$, then $s(\alpha, M) \neq \emptyset$.*

Proof. Follows immediately from the construction of $\Pi^k(D, \Gamma)$ since $possible(a_j, \alpha_{j-1})$ belongs to the body of every rule/default that generates fluent of the form $holds(f(\alpha_j))$ in M. As such, $s(\alpha, M) \neq \emptyset$ means that $possible(a_j, \alpha_{j-1}) \in M$ for every $j \leq l$. This implies the conclusion of the lemma. □

Before we prove the theorems, we simplify the program $\Pi^k(D, \Gamma)$ and $\Pi^\alpha(D, \Gamma)$ and introduce some notation that will be used subsequently. For an action theory (D, Γ), let s_0 be its initial state. Define,

$$S_\alpha = R_\alpha \cup \{holds(f, []) \mid f \in s_0\}$$

and

$$S_k = \bigcup_{|\alpha| \le k} R_\alpha \cup \{holds(f, []) \mid f \in s_0\}.$$

First, we simplify $\Pi^k(D, \Gamma)$ and $\Pi^\alpha(D, \Gamma)$ by

1. removing rules of the form (12)-(13); and
2. replacing $hold([L_1, \ldots, L_n])$ with the sequence $holds(L_1), \ldots, holds(L_n)$ in every rule of the remaining program whose body contains $hold([L_1, \ldots, L_n])$.

Let us denote the new programs by π^k and π^α, respectively. It follows from Lemma 3 of [11] that each answer set of π^k (resp. π^α) corresponds to an answer set of $\Pi^k(D, \Gamma)$ (resp. $\Pi^\alpha(D, \Gamma)$) which contains the same set of literals of the form $holds(L)$, $defeated(D)$, and S_k (resp. S_α) and vice versa.

Theorem 1 *For every consistent and complete action theory (D, Γ), the program $\Pi^k(D, \Gamma)$ is consistent.*

Proof. Our discussion shows that to prove the theorem, it is enough to prove that π^k has a consistent answer set. Let L_i be the set of literals of the program π^i. It is easy to see that L_i is a splitting set of π^k if $i \le k$. Thus, the sequence $\langle L_i \rangle_{i \le k}$, is a splitting sequence of π^k. By the splitting sequence theorem [18] (included in Appendix B as Theorem 12), M^k is an answer set of π^k iff $M^k = \bigcup_{i \le k} A^i$ where $\langle A^i \rangle_{i \le k}$ is a solution to π^k with respect to $\langle L_j \rangle_{i \le k}$. Hence, to prove the theorem, we will construct a solution $\langle A^i \rangle_{i \le k}$. We prove this by induction over k.

- **Base:** The theorem is trivial for $k = 0$ since $\pi^0 = \pi^{[]}$ and this program has only facts that is consistent because of the consistency of (D, Γ).
- **Step:** Assume that we have proved the theorem for k. We need to prove it for $k+1$. It is easy to see that π^{k+1} can be splitted by $lit(\pi^k)$, the set of literals of π^k, and $\pi^k = b_{lit(\pi^k)}(\pi^{k+1})$. This implies that M is an answer set of π^{k+1} iff $M = M^k \cup M'$ where (M^k, M') is a solution of π^{k+1} with respect to $lit(\pi^k)$. That is, M^k is an answer set of π^k and M' is an answer set of $\pi' = e_{lit(\pi^k)}(\pi^{k+1} \setminus b_{lit(\pi^k)}(\pi^{k+1}), M^k)$.

  By inductive hypothesis, $M^k = \bigcup_{i \le k} A^i$ for some solution to π^k with respect to $\langle L_j \rangle_{i \le k}$. We construct an answer set A^{k+1} of π' as follows.

  – A^{k+1} contains the set $S_{k+1} \setminus S_k$,

  – For each action sequence α and action a such that $|\alpha| = k$ and a is executable in $s(\alpha, M^k)$, we select an arbitrary but unique state $s' \in \Phi(a, s(\alpha, M^k))$ and add to A^{k+1} the following literals:

- $*$ $holds(possible(a, \alpha))$,
- $*$ $holds(f(\alpha \circ a))$ where $f \in s'$,
- $*$ $defeated(inertial(f, a, \alpha))$ if $f \in s(\alpha, M^k)$ and $\bar{f} \in s'$ where $\bar{f}$ denotes the negation of f, and
- $*$ $defeated(def(f, a, \alpha))$ if $f \in s(\alpha, M^k)$ and $\bar{f} \in s'$.

By construction and the assumption that (D, Γ) is consistent, we can easily check that A^{k+1} is indeed an answer set of π'. This concludes the proof of the inductive step, and hence, the theorem.

$\square$

Theorem 2 *Let (D, Γ) be a consistent and complete action theory and α be a sequence of actions with $|\alpha| \leq k$. The following results hold:*

- *If M is an answer set of $\Pi^k(D, \Gamma)$ then $M^\alpha = M \cap lit(\Pi^\alpha(D, \Gamma))$ is an answer set of $\Pi^\alpha(D, \Gamma)$.*
- *If M^α is an answer set of $\Pi^\alpha(D, \Gamma)$ then there exists an answer set M of $\Pi^k(D, \Gamma)$ such that $M^\alpha = M \cap lit(\Pi^\alpha(D, \Gamma))$.*

Proof. It is easy to see that S_k is a splitting set of π^k and $b_{R_k}(\pi^k) = S_k$. Furthermore, let $\pi_1 = e_{S_k}(\pi^k \setminus S_k, S_k)$. Applying the splitting set theorem, we have that π_1 consists of rules of the form

$$holds(L) \leftarrow holds(L_1), \ldots, holds(L_n). \tag{43}$$

$$holds(L) \leftarrow holds(L_1), \ldots, holds(L_n), not\ defeated(D). \tag{44}$$

$$defeated(D) \leftarrow holds(\neg L). \tag{45}$$

where

- for each rule of the form (43), there exists a rule $rule(R, L, [L_1, \ldots, L_n]) \in S_k$;
- for each rule of the form (44), there exists a default $default(D, L, [L_1, \ldots, L_n]) \in S_k$; and
- for each rule of the form (45), there exists a default $default(D, L, [L_1, \ldots, L_n]) \in S_k$.

Similar arguments hold for π^α and S_α. Let $\pi_2 = e_{S_\alpha}(\pi^\alpha \setminus S_\alpha, S_\alpha)$. We have that π_2 also consists of rules of the form (43)-(45) in which the conditions following the definition of π_1 is applied to S_α. It follows from the splitting theorem and the fact that $S_\alpha \subseteq S_k$ that it suffices to prove the following:

- If M_1 is an answer set of π_1 then $M_2 = M_1 \cap lit(\pi_2)$ is an answer set of π_2.
- For each answer set M_2 of π_2 there exists an answer set M_1 of π_1 such that $M_2 = M_1 \cap lit(\pi_2)$.

We will now prove these claims.

- Let M_1 be an answer set of π_1. We will show that $M_2 = M_1 \cap lit(\pi_2)$ is an answer set of π_2. Let $\pi = (\pi_2)^{M_2}$. Because $\pi_2 \subseteq \pi_1$, we have that $\pi \subseteq (\pi_1)^{M_1}$ and therefore M_2 satisfies π. To complete the proof, we need to show the minimality of M_2. Assume the contrary, there exists a set $X \subset M_2$ and X satisfies π. We will show that $Y = X \cup (M_1 \setminus M_2)$ satisfies $(\pi_1)^{M_1}$. Consider a rule $r \in (\pi_1)^{M_1}$ whose body is satisfied by Y. Clearly, if the head of r does not belong to $lit(\pi_2)$ then it belongs to $M_2 \setminus M_1$ and therefore, r is satisfied by Y. Because of the construction of π_1 and π_2, it is easy to see that if the head of r belongs to $lit(\pi_2)$ then so is the body of r. As such, r is satisfied by X. In both cases, r is satisfied by Y. This implies that Y is a proper subset of M_1 and satisfies all the rules in $(\pi_1)^{M_1}$. This contradicts the fact that M_1 is an answer set of π_1. In other words, we have that M_2 is an answer set of π_2.

- Let M_2 be an answer set of π_2. Let π be the program obtained from $\pi_1 \setminus \pi_2$ by

  - Removing all the rules whose body contains some literals in $lit(\pi_2) \setminus M_2$.
  - Removing all the literals in M_2 from the remaining rules.

  It is easy to see that $lit(\pi) \cap lit(\pi_2) = \emptyset$ and if X is an answer set of π then $X \cup M_2$ is an answer set of π_1. Obviously, $M_1 = X \cup M_2$ is an answer set of π_1 satisfying the condition that $M_2 = M_1 \cap lit(\pi_2)$.

The theorem is proved. □

Theorem 3 *Let (D, Γ) be a consistent and complete action theory, M be an answer set of $\Pi^k(D, \Gamma)$, α be a sequence of actions with $|\alpha| \leq k$, and a be an action such that $s(\alpha, M) \neq \emptyset$ and $s(\alpha \circ a, M) \neq \emptyset$. Then, $s(\alpha \circ a, M) \in \Phi(a, s(\alpha, M))$.*

Proof. Let us assume that $\alpha = a_1, \ldots, a_l$ for some $l \leq k$. By Theorem 2, we know that there exists an answer set M^α of π^α such that $M^\alpha \subseteq M$. It is easy to see that $s(\alpha_j, M) = s(\alpha_j, M^\alpha)$ for every j. We prove by induction over i the following conclusions:

(a) $s(\alpha_i, M^\alpha)$ is a state of D if $s(\alpha_i, M^\alpha) \neq \emptyset$; and

(b) $s(\alpha_{i+1}, M^\alpha) \in \Phi(a, s(\alpha_i, M^\alpha))$ if $s(\alpha_{i+1}, M^\alpha) \neq \emptyset$.

- **Base:** $i = 0$. We have that $s([], M) = \{f \mid$ " **initially** f" $\in \Gamma\}$. Because of the completeness and consistency of (D, Γ), (a) holds. The proof that (b) holds is similar to the proof of (b) in the inductive step and is omitted here for brevity. The base case is proved.

- **Step:** Assume that we have proved (a)-(b) for $t < i$. We prove them for $t = i$. We begin with (a). Clearly, by induction hypothesis, if $s(\alpha_i, M^\alpha) \neq \emptyset$ then $s(\alpha_i, M^\alpha) \in \Phi(a_i, s(\alpha_{i-1}, M^\alpha))$ and hence $s(\alpha_i, M^\alpha)$ is a state of D. Let $u = s(\alpha_i, M^\alpha)$. Consider a fluent literal $f \in u$. First, we prove that $s(\alpha_{i+1}, M^\alpha)$ is complete. It follows from Lemma 2 that $possible(a_{i+1}, \alpha_i) \in M$. We then prove that it is a state belonging to $\Phi(a_{i+1}, s(\alpha_i, M^\alpha))$. We have two cases:

1. f is a non-inertial fluent literal. Then, the default

$$default(def(f, \alpha_{i+1}), f(\alpha_{i+1}), [possible(a_{i+1}, \alpha_i)])$$

is applicable. Thus, if $\neg f \notin s(\alpha_{i+1}, M^\alpha)$ then $f \in s(\alpha_{i+1}, M^\alpha)$.

2. f is an inertial fluent literal. Clearly, the default

$$default(inertial(a_{i+1}, f, \alpha_i), f(\alpha_{i+1}), [f(\alpha_i), possible(a_{i+1}, \alpha_i)])$$

is applicable with respect to M^α. Thus, if it is not defeated, then $f(\alpha_{i+1})$ would belong to M^α. Otherwise, the default is defeated, and hence, we have that $holds(\neg f(\alpha_{i+1})) \in M^\alpha$. Thus $f \in s(\alpha_{i+1}, M^\alpha)$ or $\neg f \in s(\alpha_{i+1}, M^\alpha)$.

The above two cases, together with the completeness of u, conclude the completeness of $s(\alpha_{i+1}, M^\alpha)$.

We now show that $s(\alpha_{i+1}, M^\alpha)$ is a state belonging to $\Phi(a_{i+1}, s(\alpha_i, M^\alpha))$. Consider an atom $h = holds(f(\alpha_{i+1}))$. We know that $h \in M^\alpha$ iff one of the following cases happens:

1. A rule of the form (16) is used in conjunction with a rule of the form (10) in deriving h. This implies that $f \in E(a_{i+1}, s(\alpha_i, M^\alpha))$.
2. A rule of the form (20) is used in conjunction with rule (11) in deriving h. This implies that $f \in s(\alpha_i, M) \cap s(\alpha_{i+1}, M^\alpha) \cap (\mathbf{F_I} \cup \overline{\mathbf{F_I}})$.
3. A rule of the form (19) is used in conjunction with a rule of the form (11) in deriving h. This implies that f is a non-inertial fluent and $f \in Def(D)$ and $\neg f \notin E(a_{i+1}, s(\alpha_i, M^\alpha))$. In other words, $f \in Def(D) \setminus E(a_{i+1}, s(\alpha_i, M^\alpha))$.
4. A rule of the form (18) is used in conjunction with a rule of the form (10) in deriving h. This implies that
$f \in Cl(E(a_{i+1}, s(\alpha_i, M^\alpha)) \cup (s(\alpha_i, M^\alpha) \cap s(\alpha_{i+1}, M^\alpha) \cap (\mathbf{F_I} \cup \overline{\mathbf{F_I}})) \cup (Def(D) \setminus E(a_{i+1}, s(\alpha_i, M^\alpha)))$.

The above four cases show that $s(\alpha_{i+1}, M^\alpha)$ is complete and is a state of D and also belongs to $\Phi(a_{i+1}, s(\alpha_i, M^\alpha))$. The theorem is proved.

□

Theorem 4 *Let (D, Γ) be a consistent and complete action theory. Then, for every sequence of actions $\alpha = [a_1, \ldots, a_k]$ and a trajectory $s_0 a_1 s_1 \ldots a_k s_k$ of (D, Γ), there exists an answer set M of $\Pi^k(D, \Gamma)$ such that $s_i = s(\alpha_i, M)$.*

Proof. It follows from Theorem 2 that it is enough to show that there exists an answer set M^α of π^α such that $s_i = s(\alpha_i, M^\alpha)$. We prove by induction over k.

- **Base:** It is easy to see that $\pi^{[]}$ has only one answer set consisting of the set

$$X = \{holds(f([])) \mid f \in s_0\}.$$

The conclusion follows immediately from the construction of X.

- **Step:** Assume that we have proved the theorem for $|\alpha| < k$. We prove it for $|\alpha| = k$. Let $L = lit(\pi^{\alpha_{k-1}})$. Since L is a splitting set of π^α, A is an answer set of π^α iff $A = A_1 \cup A_2$ and A_1 is an answer set of $b_L(\pi^\alpha)$ and A_2 is an answer set of $\pi' = e_L(\pi^\alpha \setminus b_L(\pi^\alpha), A_1)$.

  By inductive hypothesis, we can find A_1 such that $s_i = s(\alpha_i, A_1)$. It remains to be shown that we can find an answer set A_2 of π' such that $s_k = s(\alpha, A_1 \cup A_2)$. Observe that due to the fact that π' does not contain any rule with atom of the form $prefer(D, D_1)$ in the head, we can remove the rule (15) from π' without affecting its answer sets. Thus, in what follows we will omit (15) from π'. We have that $\pi' = Y \cup \pi_1$ where

  - Y is the union of the set of facts of the form (16)-(20), which belong to π^α. In other words, Y consists of rules of the form (16)-(20) in which $\beta \circ a = \alpha$.
  - π_1 is defined as follows.
    - For each atom of the form (16) in Y, the rule

      $$holds(f(\alpha)) \quad \leftarrow \quad rule(dynamic(a_k, f, \alpha_{k-1}), f(\alpha), Body). \tag{46}$$

      belongs to π_1, if $Body$ is satisfied by A_1, i.e., , $holds(a) \in A_1$ for every literal $a \in Body$.
    - For each atom of the form (17) in Y, the rule

      $$holds(L) \leftarrow rule(R, L, Body). \tag{47}$$

      belongs to π_1, if $Body$ is satisfied by A_1
    - For each atom of the form (18) in Y, the rule

      $$\begin{aligned} holds(L) \quad \leftarrow \quad & rule(D, L, [B_1, \ldots, B_n]), \\ & holds(B_1), \ldots, holds(B_n). \end{aligned} \tag{48}$$

      belongs to π_1.
    - For each atom of the form (20) in Y, the rule

      $$defeated(D) \leftarrow default(D, L, Body), holds(\neg L). \tag{49}$$

      belongs to π_1 and if $Body$ is satisfied by A_1, the rule

      $$holds(L) \quad \leftarrow \quad default(D, L, Body), not\ defeated(D). \tag{50}$$

      belongs to π_1.
    - For each atom of the form (19) in Y, the rule

      $$defeated(D) \leftarrow default(D, L, Body), holds(\neg L). \tag{51}$$

      and the rule

      $$holds(L) \quad \leftarrow \quad default(D, L, Body), not\ defeated(D). \tag{52}$$

      belongs to π_1.

We construct $A_2 = Y \cup X \cup Z$ as follows:

- $X = \{holds(f(\alpha)) \mid f \in s_k\} \cup \{holds(possible(a_k, \alpha_{k-1}))\}$ if there exists an executable condition

  $$a_k \textbf{ executable\_if } q_1, \ldots, q_m$$

  and $q_i \in s_{k-1}$.
- Z consists of
  * $defeated(inertial(f, a_k, \alpha_{k-1}))$ if f is an inertial fluent, $holds(f(\alpha_{k-1})) \in X$, and $\overline{f} \in s_{k-1}$ where $\overline{f}$ denotes the contrary of the fluent literal f, i.e., for a fluent f, $\overline{f} = \neg f$, and $\overline{\neg f} = f$; and
  * $defeated(def(f, a_k, \alpha_{k-1}))$ if f is a non-inertial fluent and $holds(f(\alpha)) \in X$ and $\overline{f} \in s_{k-1}$.

We will show that A_2 is an answer set of π' by proving that it is a minimal set that is closed under the rule of $\pi'' = (\pi')^{A_2}$. First, we begin with the closeness.

It is easy to see that A_2 is closed under the rules (16)-(20) because of the construction of Y which contains all of the rules in this form which belong to π'. A_2 is closed under the rules of the form (49) and (51) (because of the construction of Z). This shows that A_2 is closed under the rules (16)-(20), (49), and (51). Let r be a rule of π'. Consider the remaining cases:

- r is of the form (46) whose body is satisfied by A_2. Then, we have that Y contains

  $$rule(dynamic(f, a_k, \alpha_{k-1}), f(\alpha), [p_1(\alpha_{k-1}), \ldots, p_n(\alpha_{k-1}), possible(a_k, \alpha_{k-1})])$$

  and $holds(p_j(\alpha_{k-1})) \in X$ for $1 \leq j \leq m$ and $possible(a_k, \alpha_{k-1}) \in A_2$. This means that a_k is executable in s_{k-1} and $f \in s_k$. By construction of X, we have that $holds(f(\alpha)) \in A_2$. Thus, A_2 is closed under rules of the form (46) of π''.
- r is of the form (47). Then, Y contains

  $$rule(executable(a_k, \alpha_{k-1}), possible(a_k, \alpha_{k-1}), [p_1(\alpha_{k-1}), \ldots, p_n(\alpha_{k-1})])$$

  and $holds(p_j(\alpha_{k-1})) \in X$ for $1 \leq j \leq m$. Obviously, the construction of X makes sure that $holds(possible(a_k, \alpha_{k-1})) \in A_2$, i.e., A_2 is closed under rules of the form (47) of π''.
- r is of the form (48). Then, Y contains

  $$rule(causal(f, a_k, \alpha_{k-1}), f(\alpha), [p_1(\alpha), \ldots, p_n(\alpha), possible(a_k, \alpha_{k-1})])$$

  and $holds(p_j(\alpha)) \in X$ for $1 \leq j \leq n$. Thus, $f \in s_k$ and hence we have that $holds(f(\alpha)) \in X$. This implies that A_2 is closed under rules of the form (48) of π''.

- r is of the form (50). Then, A_2 contains

  $$default(inertial(f, a_k, \alpha_{k-1}), f(\alpha), [f(\alpha_{k-1}), possible(a_k, \alpha_{k-1})])$$

  and $holds(f(\alpha_{k-1}))$, does not contain $defeated(inertial(f, a_k, \alpha_{k-1}))$, and $holds(\neg f(\alpha))$. This implies that f is an inertial fluent literal and $f \in s_{k-1}$ and $\neg f \notin s_k$. Hence, $f \in s_k$. So, $holds(f(\alpha)) \in X$ and therefore A_2 is closed under rules of the form (50) of π''.
- r is of the from (52). That means that A_2 contains

  $$default(def(f, a_k, \alpha_{k-1}), f(\alpha), [possible(a_k, \alpha_{k-1})])$$

  and $holds(f(\alpha))$, and does not contain $defeated(def(f, a_k, \alpha_{k-1}))$ and $holds(\neg f(\alpha))$. This implies that f is a non-inertial fluent and $\neg f \notin s_k$. Thus, A_2 is closed under the rules of the form (50) of π''.

The above discussion shows that A_2 is closed under the rules of π''. To complete the proof, we need to show that no proper subset of A_2 is closed under the rules of π''. Assume the contrary, there exists $B \subset A_2$ and B is closed under the rules of π''. Consider some literal $h \in A_2 \setminus B$. Obviously, h cannot be a fact of π'' which is either a literal of the form $rule(R, L, B)$ or $default(D, L, B)$. Consider the two cases:

- h is of the form $holds(f(\alpha))$ for some fluent literal $f \in s_k$. There are four cases: $h \in E(a_k, s_{k-1})$, $h \in s_k \cap s_{k-1} \cap (\mathbf{F_I} \cup \overline{\mathbf{F_I}})$, $h \in Def(D) \setminus E(a_k, s_{k-1})$, or $h \notin E(a_k, s_k) \cup (s_k \cap s_{k-1} \cap (\mathbf{F_I}) \cup \overline{\mathbf{F_I}}) \cup (Def(D) \setminus E(a_k, s_{k-1}))$. The first case cannot happen because π'' contains a rule of the form (46) whose body is satisfied by A_2 and whose head is h. The second and third case cannot happen because π'' contains a rule of the form (50) and (52), respectively, whose body is satisfied by A_2 and whose head is h. Finally, the fourth case cannot happen because π'' contains a sequence of rules of the form (48), say $r_1, \ldots, r_t$, where $body(r_1) \subseteq A_2 \cap B$ and $body(r_{j+1}) \subseteq (A_2 \cap B) \cup \{head(r_l) \mid 1 \leq l \leq j\}$.
- h is of the form $defeated(inertial(f, a_k, \alpha_{k-1}))$. This can happen only if there exists some literal $holds(\neg f(\alpha))$ in $A_2 \setminus B$. The first case shows that this cannot happen.
- h is of the form $defeated(def(f, a_k, \alpha_{k-1}))$. This can happen only if there exists some literal $holds(\neg f(\alpha))$ in $A_2 \setminus B$. The first case shows that this cannot happen.
- h is of the form $holds(possible(a_k, \alpha_{k-1}))$. This cannot happen since a_k is executable in s_{k-1}.

The above cases show that A_2 is minimal set closed under π''. Together with the closeness of A_2 under the rules of π'' we have that A_2 is an answer set of π'. This proves the inductive step since $s_k = s(\alpha, X) = s(\alpha, A_2)$.

□

Before we prove the other theorems, we prove some lemmas that will be used in the proof of Theorem 8.

Lemma 3 *Let Q be a logic program. Let Q^+ be the program obtained from Q by replacing each literal l in Q by a new and distinguished literal l^+ that does not belong to the language of Q. Then, M is an answer set of Q iff M^+ is an answer set of Q^+ where $M^+ = \{l^+ \mid l \in M\}$.*

Proof. The conclusion of the lemma follows from the equation $(Q^+)^{(M^+)} = (Q^M)^+$. (Q^M denotes the reduct of a program P with respect to the set of literals M.) □

Lemma 4 *Let Q be a logic program. Let $Q_1 = \{l_1, \ldots, l_n\}$ be a set of facts in Q and $Q_2 = \{q_1, \ldots, q_m\}$ be a set of new atoms that do not occur in Q. Let R be the set of rules $\{l_i \leftarrow q_j \mid i = 1, \ldots, n\}$ and $Q^+ = R \cup (Q \setminus Q_1)$. Then, M is an answer set of Q iff $M^+ = M \cup \{q \mid q \in Q_2$, and R contains a rule whose body is $q\}$ is an answer set of Q^+.*

Proof. Because of $Q_2 \cap lit(Q) = \emptyset$, Q_2 is a splitting set of Q^+. Furthermore, $b_{Q_2}(Q^+) = Q_2$ that has a unique answer set Q_2. Thus, M^+ is an answer set of Q^+ iff $M^+ = M \cup Q_2$ where M is an answer set of $e_{Q_2}(Q^+ \setminus b_{Q_2}(Q^+), Q_2) = Q$. □

In the next lemma we prove the correctness of an SMODELS encoding of prioritized default theory. Let T be a prioritized default theory with the underlying propositional and finite language $\mathcal{L}$, i.e., T is a finite set of ground literals of the form (7)-(9). By definition, $P(T)$ is the program consisting of T and the set of rules (10)-(15). For each list $[l_1, \ldots, l_n]$, we associate to its a new and distinguished name $n_{[l_1,\ldots,l_n]}$. The SMODELS-encoding of T, denoted by $sm(T)$, consists of the following:

- The prioritized default theory T^+ that consists of the following literals:
  - Each atom $rule(r, l_0, [l_1, \ldots, l_m])$ in T is translated into an atom
  $$rule(r, l_0, n_{[l_1,\ldots,l_m]})$$
  of T^+;
  - Each atom $default(d, l_0, [l_1, \ldots, l_m])$ in T is translated into a rule
  $$default(d, l_0, n_{[l_1,\ldots,l_m]})$$
  of T^+;
  - Each atom $prefer(d_1, d_2)$ in T is translated into a rule
  $$prefer(d_1, d_2)$$
  of T^+;
- The rules (10)-(11) and (14)-(15) from $P(T)$; and
- For each new name $n[l_1, \ldots, l_n]$ in $sm(T)$, the rule
  $$hold(n_{[l_1,\ldots,l_n]}) \leftarrow holds(l_1), \ldots, holds(l_n) \tag{53}$$
  belongs to $sm(T)$.

Lemma 5 *For a prioritized default theory T,*

- *if M is an answer set of $P(T)$ then there exists an answer set M' of $sm(T)$ such that $holds(l) \in M$ iff $holds(l) \in M'$; and*
- *if M is an answer set of $sm(T)$ then there exists an answer set M' of $P(T)$ such that $holds(l) \in M$ iff $holds(l) \in M'$.*

Proof.

- Let M be an answer set of $P(T)$ and $lit(hold)$ be the set of literals of the form $hold(L)$ in $lit(P(T))$. By $name(P(T))$ we denote the set of names introduced in creating $sm(T)$. Let Q be the set of literals of the form $hold(n_{[l_1,\ldots,l_n]}) \in M$ such that $n_{[l_1,\ldots,l_n]} \in name(P(T))$ and $\{holds(l_i) \mid i = 1, \ldots, n\} \subseteq M$. We define

  $$M' = M \setminus (lit(hold) \cup T) \cup T^+ \cup Q$$

  and prove that M' is an answer set of $sm(T)$. We prove this by showing that M' is a minimal set of literals in $lit(sm(T))$ that satisfies the rules in $(sm(T))^{M'}$.

  – *Satisfiability*: It is easy to see that each rule r^+, except the rule of the form (53), in $(sm(T))^{M'}$ corresponds to a rule r in $(P(T))^M$ and if $body(r^+) \subseteq M'$ then $body(r) \subseteq M$. Thus, $head(r) \in M$ since M is an answer set of $P(T)$. The construction of M' implies that $head(r^+) \in M'$. Furthermore, because of the construction of Q, M' satisfies all the rules of the form (53) of $(sm(T))^{M'}$. So, M' satisfies $(sm(T))^{M'}$.

  – *Minimality:* Assume the contrary, M' is not minimal, i.e., there exists $M'' \subset M'$, $M' \setminus M'' \neq \emptyset$, and M'' satisfies all the rules of $(sm(T))^{M'}$. Consider $l \in M'' \setminus M'$. From the construction of M' and $sm(T)$, we conclude that l is of the form $holds(l')$. It follows that there exists some rule r in $(P(T))^M$ such that $head(r) = l$ and $body(r) \subseteq M$. Consider the rule r^+ of $sm(T)$ that corresponds to r. Clearly, $body(r^+) \subseteq M'$. This implies that M'' does not satisfy r^+, i.e., M'' does not satisfy $(sm(T))^{M'}$, which contradicts our assumption. This contradiction implies that M' is a minimal set of literals that is satisfies $(sm(T))^{M'}$.

  The above two properties show that M' is an answer set of $sm(T)$. The construction of M' ensures that $holds(l) \in M$ iff $holds(l) \in M'$.

- Let M be an answer set of $sm(T)$. By $name(P(T))$ we denote the set of names introduced in creating $sm(T)$. Let Q_1 be the set of literals of the form $holds(n_{[l_1,\ldots,l_n]}) \in M$. Let Q_2 be the set of all literals of the form $hold([l_1, \ldots, l_n])$ such that $l_i \notin name(P(T))$ and $\{holds(l_i) \mid i = 1, \ldots, n\} \subseteq M$. We define

  $$M' = M \setminus (Q_1 \cup T^+) \cup T \cup Q_2$$

  and prove that M' is an answer set of $P(T)$. We prove this by showing that M' is a minimal set of literals in $P(T)$ that is closed under $(P(T))^{M'}$.

– *Closeness*: Consider a rule r in $(P(T))^{M'}$. From the construction of $sm(T)$, we can conclude that if r is of the form (10)-(11) or (14)-(15) then there exists a rule r^+ in $(sm(T))^M$ such that if $body(r) \subseteq M$ then $body(r^+) \subseteq M'$. Thus, $head(r) \in M$. The construction of M' implies that $head(r^+) \in M'$. This implies that M' is satisfies all the rules of the form (10)-(11) or (14)-(15) in $(P(T))^{M'}$. The construction of Q_2 ensures that M' satisfies all the rules of the form (12)-(13) of $(P(T))^{M'}$. So, M' satisfies $(P(T))^{M'}$.

– *Minimality:* Assume the contrary, M' is not minimal, i.e., there exists $M'' \subsetneq M'$ and M'' satisfies all the rules of $(P(T))^{M'}$. Consider $l \in M'' \setminus M'$. From the construction of M' and the definition of $P(T)$, we conclude that l is of the form $holds(l')$. It follows that there exists some rule r^+ in $(sm(T))^M$ such that $head(r) = l$ and $body(r) \subseteq M$. Let r be the rule from which r^+ is constructed. We have that $body(r) \subseteq M'$ and $head(r) \not\in M''$, which implies that M'' does not satisfy r, i.e., M'' does not satisfy $(P(T))^{M'}$, which contradicts our assumption. This contradiction implies that M' is a minimal set of literals satisfying $(P(T))^{M'}$.

The above two properties show that M' is an answer set of $P(T)$. The construction of M' ensures that $holds(l) \in M$ iff $holds(l) \in M'$.

The proof of the lemma follows from the above two cases. $\square$

We will use the above three lemmas to prove Theorem 8. First, for each literal $l \in lit(P^\alpha)$ we define l^+ as follows.

1. if $l = rule(dynamic(f, a, \beta), f(\beta \circ a), [p_1(\beta), \ldots, p_n(\beta), possible(a, \beta)])$
then $l^+ = rule(dynamic(f, a, k), f(k+1), [p_1(k), \ldots, p_n(k), possible(a, k)])$
where $|\beta| = k$;

2. if $l = rule(causal(f, a, \beta), f(\beta \circ a), [p_1(\beta \circ a), \ldots, p_n(\beta \circ a), possible(a, \beta)])$
then $l^+ = rule(dynamic(f, a, k), f(k+1), [p_1(k+1), \ldots, p_n(k+1), possible(a, k)])$
where $|\beta| = k$;

3. if $l = default(def(f, a, \beta), f(\beta \circ a), [possible(a, \beta)])$
then $l^+ = default(def(f, a, k), f(k+1), [possible(a, k)])$ where $|\beta| = k$;

4. if $l = default(inertial(f, a, \beta), f(\beta \circ a), [f(\beta), possible(a, \beta)])$
then $l^+ = default(inertial(f, a, k), f(k+1), [f(k), possible(a, k)])$ where $|\beta| = k$;

5. if $l = rule(executable(a, \beta), possible(a, \beta), [p_1(\beta), \ldots, p_n(\beta)])$
then $l^+ = rule(dynamic(a, k), f(k), [p_1(k), \ldots, p_n(k)])$ where $|\beta| = k$;

6. if $l = holds(f(\beta))$ then $l^+ = holds(f(k))$ where $|\beta| = k$;

7. if $l = holds(possible(a, \beta))$ then $l^+ = holds(possible(a, k))$ where $|\beta| = k$;

8. if $l = defeated(def(f, a, \beta))$ then $l^+ = defeated(def(f, a, k))$ where $|\beta| = k$; and

9. if $l = defeated(inertial(f, a, \beta))$ then $l^+ = defeated(inertial(f, a, k))$ where $|\beta| = k$.

Let α be a sequence of actions $a_1, \ldots, a_k$. Let $Q_\alpha = (P^\alpha)^+$ where l^+ is defined for each literal l in $lit(P^\alpha)$ as above. Let Q_1 be the set of facts $\{l_1, \ldots, l_k\}$ of Q where l_j is of the form (1)-(4) as described above (i.e., l_j is a rule or default constructed from (D, Γ) except those correspond to executability conditions of (D, Γ)). Furthermore, let Q_2 is the set of facts $\{occ(a_i, i-1) \mid i = 1, \ldots, k\}$. It follows from the Lemmas 3-4 that M is an answer set of P^α iff $M^+ \cup Q_2$ is an answer set of $(Q_\alpha)^+ = Q_\alpha \cup Q_2$. Let Q_3 be the program obtained from $(Q_\alpha)^+$ by:

- replacing each occurrence of a list $[l_1, \ldots, l_k]$ in the atoms of the form $rule(.,.,.)$, $default(.,.,.)$ with the atom associated to the list as in the translation from P^α to Q_α in each rule or default in $(Q_\alpha)^+$;
- removing the rules (12)-(13); and
- adding a rule
$$hold(n_{[l_1,\ldots,l_n]}) \leftarrow holds(l_1), \ldots, holds(l_k)$$
for each new name $n_{[l_1,\ldots,l_n]}$.

We can prove the following lemma.

Lemma 6 *Let (D, Γ) be a consistent and complete an action theory and $\alpha = a_1, \ldots, a_n$ be a sequence of actions. Then,*

1. *if M is an answer set of $P^\alpha(D, \Gamma)$, then Q_3 has an answer set M' with the property that*
$$M \models holds(f(a_1 \circ \cdots \circ a_i)) \text{ if and only if } M' \models holds(f(i))$$
2. *if M is an answer set of Q_3 then there exists an answer set M' of $P^\alpha(D, \Gamma)$ such that*
$$M \models holds(f(i)) \text{ if and only if } M' \models holds(f(a_1 \circ \cdots \circ a_i])).$$

Proof. The proof is based on Lemmas 3–5. First, P^α is translated into a program Q_α in which every occurrence of an action sequence β is replaced by $|\beta|$. Second, to create $(Q_\alpha)^+$ from Q_α, the set of action occurrences $Q_2 = \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$ is introduced to Q_α with respect to the set of rules Q_1, which consist of all the rules and defaults, whose body does not contain literals of the from (A.ii), of the prioritized default theory corresponding to (D, Γ). Third, the list notation is dropped by introducing the names for lists and adding the rules (53) to create Q_3.

- Let M be an answer set of P^α. Lemma 3 implies that there exists an answer set M_1 of Q_α such that $holds(f(a_1 \circ \cdots \circ a_i)) \in M$ iff $holds(f(i)) \in M_1$. It follows from Lemma 4 that there exists an answer set M_2 of $(Q_\alpha)^+$ such that $holds(f(i)) \in M_1$ iff $holds(f(i)) \in M_2$. Lemma 5 implies that there exists an answer set M' of Q_3 such that $holds(f(i)) \in M_2$ iff $holds(f(i)) \in M'$. This proves the first item of the theorem.
- The proof of the second item of the theorem is similar to the proof of the first item—based on the second item of the Lemmas 3-5.

□

We now give the proof of Theorem 8.

Theorem 8 *Let (D, Γ) be a consistent and complete action theory and $\alpha = a_1, \ldots, a_k$ be a sequence of actions. Then,*

1. *if M is an answer set of $P^\alpha(D, \Gamma)$, then $SM^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$ has an answer set M' with the property that*

$$M \models holds(f(a_1 \circ \cdots \circ a_i)) \text{ if and only if } M' \models holds(f(i))$$

2. *if M is an answer set of $SM^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$ then there exists an answer set M' of $P^\alpha(D, \Gamma)$ such that*

$$M \models holds(f(i)) \text{ if and only if } M' \models holds(f(a_1 \circ \cdots \circ a_i)).$$

Proof. Let $P = SM^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$. It follows from Lemma 6 that it suffices to prove that Q_3 and P are equivalent. For $Pred \in \{rule, default, hold, holds\}$ and $Prog \in \{P, Q_3\}$, by $lit(Prog, Pred)$ we denote the set of literals whose predicate name is $Pred$ in the program $Prog$. For a set of literals S in Q_3 (resp P), $M_1(S)$ (resp. $N_1(S)$) we denote the set of rules Q_3 whose body is empty or contains only the literal of the form $occ(a, k)$. It is easy to see that if M is an answer set of Q_3, then $M \cap (lit(Q_3, rule) \cup lit(Q_3, default)) = M_1(S)$. Similarly, if M is an answer set of P, then $M \cap (lit(P, rule) \cup lit(P, default)) = N_1(S)$. Observe that for each rule r of the form (29)-(33) in P there exists one and only one rule r' in $Q_3 \cap (lit(Q_3, default) \cup lit(Q_3, rule))$ of Q_3 with the following property: (i) r and r' have the same body; and (ii) the heads of r and r' refer to the same default or rule of the original prioritized default theory, i.e., this correspondence is one-to-one. In what follows, we will use r' to refer to rule in Q_3 and use r to refer to its correspondence in P.

We now prove the theorem.

- Let S be an answer set of Q_3. We have that $holds(possible(a_j, j-1)) \in S$ for $j = 1, \ldots, k$ (Lemma 6 and Theorem 4). Let

$$S_{pdt} = \{head(r) \mid r' \in M_1(S)\}$$

and

$$S_{hold} = \{hold(n_{[l_1, \ldots, l_m]}(t)) \mid n_{[l_1, \ldots, l_m]} \text{ is a name in } P,\ holds(l_i(t)) \in S \text{ for } i = 1, \ldots, m\}.$$

Intuitively, S_{pdt} accounts for the difference in the set of rules and defaults while S_{hold} accounts for the difference between the names in P and Q_3. We will show that

$$S' = (S \setminus (M_1(S) \cup lit(Q_3, hold))) \cup S_{pdt} \cup S_{hold}$$

is an answer set of P. It is easy to see that S' satisfies all rules of P. Suppose that there exists $S'' \subset S'$ that satisfies P^M. Consider $l \in S'' \setminus M'$. We know that l cannot have the form $occ(a_i, i-1)$ or $holds(possible(a_j, j-1))$. It can also not be in $lit(hold)$ by the construction of S'. Hence, l is of the form $holds(f(t))$ for some fluent literal f. This implies that $holds(f(t)) \in S$. From the construction of Q_3, we conclude that there exists a rule r' in $lit(Q_3, default)$ or $lit(Q_3, rule)$ whose head is $f(t)$ and whose body is satisfied by S. Consider its correspondence in P, the rule r. It is easy to see that the body of r is satisfied by S', i.e., $l \in S''$. This contradicts the fact that $l \in S' \setminus S''$. Therefore, S' is an answer set of P.

- Let S be an answer set of P. Because of the constraint (35) and $occ(a_i, i-1) \in P$, we conclude that $holds(possible(a_i, i-1)) \in S$ for $i = 1, \ldots, k$. Similarly, we can show that $S' = (S \setminus (N_1(S) \cup lit(P, hold))) \cup \{head(r') \mid r \text{ in } N_1(S)\} \cup \{hold(n_{[l_1,\ldots,l_m]}) \mid holds(l_i) \in S$ for $i = 1, \ldots, m\}$ is an answer set of Q_3.

The above two cases show that Q_3 and P are equivalent. In other words, we prove the theorem. □
Theorem 8, together with Theorem 4, yields the proof of Theorem 9.

Theorem 9 *For a consistent and complete action theory* (D, Γ).

1. *If* $s_0 a_1 \ldots a_k s_k$ *is a trajectory for* Δ *then* $SM^{Plan,k}(D, \Gamma, \Delta)$ *has an answer set* M *such that*
   (a) $occ(a_i, i-1) \in M$ *for every integer* i, $1 \leq i \leq k$, *and*
   (b) $s_i = \{f \mid holds(f(i)) \in M\}$.
2. *If* $SM^{Plan,k}(D, \Gamma, \Delta)$ *has an answer set* M *such that*
   (a) $occ(a_i, i-1) \in M$ *for every integer* i, $1 \leq i \leq k$, *and*
   (b) $s_i = \{f \mid holds(f(i)) \in M\}$

   then $s_0 a_1 \ldots a_k s_k$ *is a trajectory for* Δ.

Proof.

- Let $\alpha = a_1, \ldots, a_k$. Since $s_0 a_1 \ldots a_k s_k$ is a trajectory for Δ, by Theorem 4 there exists an answer set M' of P^α such that $s(\alpha_i, M') = s_i$. Theorem 8 implies that $SM^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$ has an answer set M such that $holds(f(\alpha_i)) \in M'$ iff $holds(f(i)) \in M$. In other words, we have that $s_i = s(\alpha_i, M') = \{holds(f(i)) \mid holds(f(i)) \in M\}$. It is easy to verify that M is indeed also an answer set of $SM^{Plan,k}(D, \Gamma, \Delta)$.

- Let $\alpha = a_1, \ldots, a_k$. It is easy to see that $M' = (M \cap lit(SM^k(D, \Gamma))) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$ is an answer set of $SM^k(D, \Gamma) \cup \{occ(a_i, i-1) \mid i = 1, \ldots, k\}$. Thus, Theorem 8 implies that there exists an answer set set M'' of P^α such that $holds(f(\alpha_i)) \in M''$ iff $holds(f(i)) \in M$. Because of the rule (37) we have that s_k satisfies the goal. This, together with Theorem 4, implies that $s_0 a_1 \ldots a_k s_k$ is indeed a trajectory achieving Δ.

The above two cases prove the theorem. □.
We now prove Theorem 10.

Theorem 10 *Let (D, Γ) be a consistent and complete action theory and M be an answer set of the program $SM^{Pref,k}(D, \Gamma, \Delta)$ encoding the planning problem (D, Γ, Δ) with a set of preferences P. Then, $s_0 a_1 s_1 \cdots a_n s_n$ is a most preferred trajectory satisfying Δ where*

- $occ(a_i, i-1) \in M$
- $s_i = \{f \mid holds(f(i)) \in M\}$

Proof. Let $tr(M) = s_0 a_1 s_1 \cdots a_k s_k$. Assume that $tr(M)$ is not a most preferred trajectory. By definition, there exists a trajectory $s_0 b_1 s'_1 \cdots b_m s'_m$ such that $prefer(b_i, a_i) \in Pref$ for some $i \leq \min\{m, k\}$ and for every integer j, $1 \leq j < i$, $prefer(a_j, b_j) \notin Pref$. Because of $prefer(b_i, a_i) \in Pref$ we have that

$$block(executable(a_i, i-1), true) \leftarrow goal(k)$$

is applicable in M. Thus, rule (24) implies that $occ(a_i, i-1) \notin M$. This contradicts the assumption that $occ(a_i, i-1) \in M$. $\square$

Appendix B – Answer Sets and Splitting Theorem

Consider a set of ground atoms A. The body of a rule r of the form (42) is satisfied by A if $\{a_{m+1}, \ldots, a_n\} \cap A = \emptyset$ and $\{a_1, \ldots, a_m\} \subseteq A$.

For a set of ground atoms A and a program Π, the *reduct* of Π with respect to A, denoted by Π^A, is the program obtained from the set of all ground instances of Π by deleting

1. each rule that has a naf-literal $not\ a$ in its body with $a \in S$, and
2. all naf-literals in the bodies of the remaining clauses.

S is an *answer set* of Π if it satisfies the following conditions.

1. If Π does not contain any naf-literal (i.e. $m = n$ in every rule of Π) then S is the smallest set of atoms that satisfies all the rules in Π.
2. If the program Π does contain some naf-literal ($m < n$ in some rule of Π), then S is an answer set of Π if S is the answer set of Π^S. (Note that Π^S does not contain naf-literals, its answer set is defined in the first item.)

For a program Π over the language $\mathcal{LP}$, a set of literals of $\mathcal{LP}$, A, is a splitting set of Π if for every rule $r \in \Pi$, r is of the form if $head(r) \in A$ then $lit(r) \subseteq A$.

Let A be a splitting set of Π. The *bottom of* Π *relative to A*, denoted by $b_A(\Pi)$, is the program consisting of all rules $r \in \Pi$ such that the head of r belongs to A.

Given a splitting set A for Π, and a set X of literals from $lit(b_A(\Pi))$, the *partial evaluation of* Π *by X with respect to A*, denoted by $e_A(\Pi, X)$, is the program obtained from Π as follows. For each rule $r \in \Pi \setminus b_A(\Pi)$ such that

1. $pos(r) \cap A \subseteq X$;
2. $neg(r) \cap A$ is disjoint from X;

there is a rule r' in $e_A(\Pi, X)$ such that

1. $head(r') = head(r)$, and
2. $pos(r') = pos(r) \setminus A$,
3. $neg(r') = neg(r) \setminus A$.

Let A be a splitting set of Π. A *solution to* Π *with respect to A* is a pair $\langle X, Y \rangle$ of set of literals satisfying the following two properties:

1. X is an answer set of $b_A(\Pi)$;
2. Y is an answer set of $e_A(\Pi \setminus b_A(\Pi), X)$;
3. $X \cup Y$ is consistent.

The splitting set theorem is as follows.

Theorem 11 (Splitting Set Theorem, [18]) *Let A be a splitting set for a program* Π. *A set A of literals is a consistent answer set of* Π *iff $A = X \cup Y$ for some solution $\langle X, Y \rangle$ to* Π *with respect to A.* $\square$

A *sequence* is a family whose index set is an initial segment of ordinals $\{\alpha \mid \alpha < \mu\}$. A sequence $\langle A_\alpha \rangle_{\alpha<\mu}$ of sets is *monotone* if $A_\alpha \subseteq A_\beta$ whenever $\alpha < \beta$, and *continuous* if, for each limit ordinal $\alpha < \mu$, $A_\alpha = \bigcup_{\gamma<\alpha} A_\gamma$.
A *splitting sequence* for a program Π is a nonempty, monotone, and continuous sequence $\langle A_\alpha \rangle_{\alpha<\mu}$ of splitting sets of Π such that $lit(\Pi) = \bigcup_{\alpha<\mu} A_\alpha$.
Let $\langle A_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence of the program Π. A *solution to* Π *with respect to A* is a sequence $\langle E_\alpha \rangle_{\alpha<\mu}$ of set of literals satisfying the following conditions.

1. E_0 is an answer set of the program $b_{A_0}(\Pi)$;
2. for any α such that $\alpha + 1 < \mu$, $E_{\alpha+1}$ is an answer set for $e_{A_\alpha}(b_{A_{\alpha+1}}(\Pi) \setminus b_{A_\alpha}(\Pi), \bigcup_{\gamma\leq\alpha} E_\gamma)$;
3. For any limit ordinal $\alpha < \mu$, $E_\alpha = \emptyset$;
4. $\bigcup_{\gamma\leq\mu} E_\gamma$ is consistent.

The splitting set theorem is generalized for splitting sequence next.

Theorem 12 (Splitting Sequence Theorem, [18]) *Let $A = \langle A_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence of the program* Π. *A set of literals E is a consistent answer set of* Π *iff $E = \bigcup_{\alpha<\mu} E_\alpha$ for some solution $\langle E_\alpha \rangle_{\alpha<\mu}$ to* Π *with respect to A.* $\square$

Appendix C – Sample Translation

In this appendix, we present the details of the translation from action theory into prioritized default theory as well as its SMODELS encoding.

Action Theory

The action theory (D, Γ) contains the following propositions:

- D contains:

  a **causes** f **if** g
  b **causes** g **if** $\neg g$
  c **causes** f **if** $\neg h$
  h **if** g

- Γ contains:

  initially $\neg f$
  initially $\neg g$
  initially $\neg h$

The formula Δ representing the goal is

$$\Delta = f$$

The goal is f and we are assuming the presence of a preference

$$\neg h \prec h$$

Prioritized Default Theory

Let us focus on sequences of actions of length at most 2. The corresponding prioritized default theory contains the following rule and default definitions. In all the rules and defaults, ϵ denotes the empty sequence of actions.

Dynamic Causal Laws

$$\begin{array}{l}
rule(dynamic(f,a,\epsilon),f(a),[g(\epsilon),possible(a,\epsilon)]) \\
rule(dynamic(f,a,a),f(aa),[g(a),possible(a,a)]) \\
rule(dynamic(f,a,b),f(ba),[g(b),possible(a,b)]) \\
rule(dynamic(f,a,c),f(ca),[g(c),possible(a,c)]) \\
\\
rule(dynamic(g,b,\epsilon),g(b),[\neg g(\epsilon),possible(b,\epsilon)]) \\
rule(dynamic(g,b,a),g(ab),[\neg g(a),possible(b,a)]) \\
rule(dynamic(g,b,b),g(bb),[\neg g(b),possible(b,b)]) \\
rule(dynamic(g,b,c),g(cb),[\neg g(c),possible(b,c)]) \\
\\
rule(dynamic(f,c,\epsilon),f(c),[\neg h(\epsilon),possible(c,\epsilon)]) \\
rule(dynamic(f,c,a),f(c),[\neg h(a),possible(c,a)]) \\
rule(dynamic(f,c,b),f(c),[\neg h(b),possible(c,b)]) \\
rule(dynamic(f,c,c),f(c),[\neg h(c),possible(c,c)])
\end{array}$$

Executability Conditions

$$\begin{array}{l}
rule(executable(a,\epsilon),possible(a,\epsilon),[]) \\
rule(executable(a,a),possible(a,a),[]) \\
rule(executable(a,b),possible(a,b),[]) \\
rule(executable(a,c),possible(a,c),[]) \\
\\
rule(executable(b,\epsilon),possible(b,\epsilon),[]) \\
rule(executable(b,a),possible(b,a),[]) \\
rule(executable(b,b),possible(b,b),[]) \\
rule(executable(b,c),possible(b,c),[]) \\
\\
rule(executable(c,\epsilon),possible(c,\epsilon),[]) \\
rule(executable(c,a),possible(c,a),[]) \\
rule(executable(c,b),possible(c,b),[]) \\
rule(executable(c,c),possible(c,c),[])
\end{array}$$

Static Causal Laws

$$\begin{array}{l}
rule(causal(h,a,\epsilon),h(a),[g(a),possible(a,\epsilon)])\\
rule(causal(h,a,a),h(aa),[g(aa),possible(a,a)])\\
rule(causal(h,a,b),h(ba),[g(ba),possible(a,b)])\\
rule(causal(h,a,c),h(ca),[g(ca),possible(a,c)])\\
\\
rule(causal(h,b,\epsilon),h(b),[g(b),possible(b,\epsilon)])\\
rule(causal(h,b,a),h(ab),[g(ab),possible(b,a)])\\
rule(causal(h,b,b),h(bb),[g(bb),possible(b,b)])\\
rule(causal(h,b,c),h(cb),[g(cb),possible(b,c)])\\
\\
rule(causal(h,c,\epsilon),h(c),[g(c),possible(c,\epsilon)])\\
rule(causal(h,c,a),h(ac),[g(ac),possible(c,a)])\\
rule(causal(h,c,b),h(bc),[g(bc),possible(c,b)])\\
rule(causal(h,c,c),h(cc),[g(cc),possible(c,c)])
\end{array}$$

Inertial Axioms

The inertial axioms are encoded as follows.

- **Inertial defaults for the fluent literal** f

$$\begin{array}{l}
default(inert(f,a,\epsilon),f(a),[f(\epsilon),possible(a,\epsilon)])\\
default(inert(f,a,a),f(aa),[f(a),possible(a,a)])\\
default(inert(f,a,b),f(ba),[f(b),possible(a,b)])\\
default(inert(f,a,c),f(ca),[f(c),possible(a,c)])\\
default(inert(f,b,\epsilon),f(b),[f(\epsilon),possible(b,\epsilon)])\\
default(inert(f,b,a),f(ab),[f(a),possible(b,a)])\\
default(inert(f,b,b),f(bb),[f(b),possible(b,b)])\\
default(inert(f,b,c),f(cb),[f(c),possible(b,c)])\\
default(inert(f,c,\epsilon),f(c),[f(\epsilon),possible(c,\epsilon)])\\
default(inert(f,c,a),f(ac),[f(a),possible(c,a)])\\
default(inert(f,c,b),f(bc),[f(b),possible(c,b)])\\
default(inert(f,c,c),f(cc),[f(c),possible(c,c)])
\end{array}$$

- **Inertial defaults for the fluent literal** g

$$\begin{array}{l}
default(inert(g,a,\epsilon),g(a),[g(\epsilon),possible(a,\epsilon)])\\
default(inert(g,a,a),g(aa),[g(a),possible(a,a)])\\
default(inert(g,a,b),g(ba),[g(b),possible(a,b)])\\
default(inert(g,a,c),g(ca),[g(c),possible(a,c)])
\end{array}$$

$$default(inert(g,b,\epsilon),g(b),[g(\epsilon),possible(b,\epsilon)])$$
$$default(inert(g,b,a),g(ab),[g(a),possible(b,a)])$$
$$default(inert(g,b,b),g(bb),[g(b),possible(b,b)])$$
$$default(inert(g,b,c),g(cb),[g(c),possible(b,c)])$$
$$default(inert(g,c,\epsilon),g(c),[g(\epsilon),possible(c,\epsilon)])$$
$$default(inert(g,c,a),g(ac),[g(a),possible(c,a)])$$
$$default(inert(g,c,b),g(bc),[g(b),possible(c,b)])$$
$$default(inert(g,c,c),g(cc),[g(c),possible(c,c)])$$

- **Inertial defaults for the fluent literal** h

$$default(inert(h,a,\epsilon),h(a),[h(\epsilon),possible(a,\epsilon)])$$
$$default(inert(h,a,a),h(aa),[h(a),possible(a,a)])$$
$$default(inert(h,a,b),h(ba),[h(b),possible(a,b)])$$
$$default(inert(h,a,c),h(ca),[h(c),possible(a,c)])$$
$$default(inert(h,b,\epsilon),h(b),[h(\epsilon),possible(b,\epsilon)])$$
$$default(inert(h,b,a),h(ab),[h(a),possible(b,a)])$$
$$default(inert(h,b,b),h(bb),[h(b),possible(b,b)])$$
$$default(inert(h,b,c),h(cb),[h(c),possible(b,c)])$$
$$default(inert(h,c,\epsilon),h(c),[h(\epsilon),possible(c,\epsilon)])$$
$$default(inert(h,c,a),h(ac),[h(a),possible(c,a)])$$
$$default(inert(h,c,b),h(bc),[h(b),possible(c,b)])$$
$$default(inert(h,c,c),h(cc),[h(c),possible(c,c)])$$

- **Inertial defaults for the fluent literal** $\neg f$

$$default(inert(\neg f,a,\epsilon),\neg f(a),[\neg f(\epsilon),possible(a,\epsilon)])$$
$$default(inert(\neg f,a,a),\neg f(aa),[\neg f(a),possible(a,a)])$$
$$default(inert(\neg f,a,b),\neg f(ba),[\neg f(b),possible(a,b)])$$
$$default(inert(\neg f,a,c),\neg f(ca),[\neg f(c),possible(a,c)])$$
$$default(inert(\neg f,b,\epsilon),\neg f(b),[\neg f(\epsilon),possible(b,\epsilon)])$$
$$default(inert(\neg f,b,a),\neg f(ab),[\neg f(a),possible(b,a)])$$
$$default(inert(\neg f,b,b),\neg f(bb),[\neg f(b),possible(b,b)])$$
$$default(inert(\neg f,b,c),\neg f(cb),[\neg f(c),possible(b,c)])$$
$$default(inert(\neg f,c,\epsilon),\neg f(c),[\neg f(\epsilon),possible(c,\epsilon)])$$
$$default(inert(\neg f,c,a),\neg f(ac),[\neg f(a),possible(c,a)])$$
$$default(inert(\neg f,c,b),\neg f(bc),[\neg f(b),possible(c,b)])$$
$$default(inert(\neg f,c,c),\neg f(cc),[\neg f(c),possible(c,c)])$$

- **Inertial defaults for the fluent literal** $\neg g$

$$\begin{array}{l}
default(inert(\neg g, a, \epsilon), \neg g(a), [\neg g(\epsilon), possible(a, \epsilon)]) \\
default(inert(\neg g, a, a), \neg g(aa), [\neg g(a), possible(a, a)]) \\
default(inert(\neg g, a, b), \neg g(ba), [\neg g(b), possible(a, b)]) \\
default(inert(\neg g, a, c), \neg g(ca), [\neg g(c), possible(a, c)]) \\
default(inert(\neg g, b, \epsilon), \neg g(b), [\neg g(\epsilon), possible(b, \epsilon)]) \\
default(inert(\neg g, b, a), \neg g(ab), [\neg g(a), possible(b, a)]) \\
default(inert(\neg g, b, b), \neg g(bb), [\neg g(b), possible(b, b)]) \\
default(inert(\neg g, b, c), \neg g(cb), [\neg g(c), possible(b, c)]) \\
default(inert(\neg g, c, \epsilon), \neg g(c), [\neg g(\epsilon), possible(c, \epsilon)]) \\
default(inert(\neg g, c, a), \neg g(ac), [\neg g(a), possible(c, a)]) \\
default(inert(\neg g, c, b), \neg g(bc), [\neg g(b), possible(c, b)]) \\
default(inert(\neg g, c, c), \neg g(cc), [\neg g(c), possible(c, c)])
\end{array}$$

- **Inertial defaults for the fluent literal** $\neg h$

$$\begin{array}{l}
default(inert(\neg h, a, \epsilon), \neg h(a), [\neg h(\epsilon), possible(a, \epsilon)]) \\
default(inert(\neg h, a, a), \neg h(aa), [\neg h(a), possible(a, a)]) \\
default(inert(\neg h, a, b), \neg h(ba), [\neg h(b), possible(a, b)]) \\
default(inert(\neg h, a, c), \neg h(ca), [\neg h(c), possible(a, c)]) \\
default(inert(\neg h, b, \epsilon), \neg h(b), [\neg h(\epsilon), possible(b, \epsilon)]) \\
default(inert(\neg h, b, a), \neg h(ab), [\neg h(a), possible(b, a)]) \\
default(inert(\neg h, b, b), \neg h(bb), [\neg h(b), possible(b, b)]) \\
default(inert(\neg h, b, c), \neg h(cb), [\neg h(c), possible(b, c)]) \\
default(inert(\neg h, c, \epsilon), \neg h(c), [\neg h(\epsilon), possible(c, \epsilon)]) \\
default(inert(\neg h, c, a), \neg h(ac), [\neg h(a), possible(c, a)]) \\
default(inert(\neg h, c, b), \neg h(bc), [\neg h(b), possible(c, b)]) \\
default(inert(\neg h, c, c), \neg h(cc), [\neg h(c), possible(c, c)])
\end{array}$$

Initial State

The initial state Γ leads to the following collection of facts:

$$\begin{array}{c}
holds(\neg f(\epsilon)) \\
holds(\neg g(\epsilon)) \\
holds(\neg h(\epsilon))
\end{array}$$

SMODELS **Encoding**

Let us illustrate the structure of the $SM^2(D, \Gamma)$ generated from this program.

Translation of D

The axioms in Γ are translated as follows:

$$\begin{array}{l} holds(neg\_f(0)). \\ holds(neg\_g(0)). \\ holds(neg\_g(0)). \end{array}$$

The dynamic causal laws are translated as follows:

$$\begin{array}{lcl} rule(dynamic(f,a,T), f(T+1), n_1(T)) & :- & time(T), occ(a,T). \\ rule(dynamic(g,b,T), g(T+1), n_2(T)) & :- & time(T), occ(b,T). \\ rule(dynamic(f,c,T), f(T+1), n_3(T)) & :- & time(T), occ(c,T). \end{array}$$

Table 1 shows the mapping between names and lists of atoms used for the encoding presented here. (Notice that we simplify the encoding by using numerical indices in the names instead of using the lists of fluent literals as indices).

List Name	*List Value*
$n_0(T)$	$[]$
$n_1(T)$	$[g(\beta_T)]$
$n_2(T)$	$[neg\_g(\beta_T)]$
$n_3(T)$	$[neg\_h(\beta_T)]$
$n_4(T)$	$[g(\beta_T)]$
$n_5(T)$	$[g(\beta_T)]$
$n_6(T)$	$[g(\beta_T)]$
$n_7(T,A)$	$[f(\beta_T)]$
$n_8(T,A)$	$[g(\beta_T)]$
$n_9(T,A)$	$[h(\beta_T)]$
$n_{10}(T,A)$	$[neg\_f(\beta_T)]$
$n_{11}(T,A)$	$[neg\_g(\beta_T)]$
$n_{12}(T,A)$	$[neg\_h(\beta_T)]$

Table 1: Encoding of Lists of Literals

The executability conditions are encoded as:

$$\begin{array}{lcl} rule(executable(a,T), possible(a,T), n_0(T)) & :- & time(T). \\ rule(executable(b,T), possible(b,T), n_0(T)) & :- & time(T). \\ rule(executable(c,T), possible(c,T), n_0(T)) & :- & time(T). \end{array}$$

The static causal laws are encoded as:

$$\begin{array}{lcl} rule(causal(h,a,T), h(T+1), n_4(T+1)) & :- & occ(a,T), time(T). \\ rule(causal(h,b,T), h(T+1), n_5(T+1)) & :- & occ(b,T), time(T). \\ rule(causal(h,c,T), h(T+1), n_6(T+1)) & :- & occ(c,T), time(T). \end{array}$$

The domain independent part of $holds$ is unchanged, and it contains the following rules:

$$\begin{array}{lll}
holds(L) & :- & rule(R, L, Body), hold(Body), not\ \ blocked(R). \\
holds(L) & :- & default(D, L, Body), hold(Body), not\ \ defeated(D). \\
blocked(R) & :- & block(R, Body), hold(Body).
\end{array}$$

The unfolding described in Section 6 transforms the $hold$ predicate as illustrated below.

$$\begin{array}{lll}
hold(n_0(T)) & :- & time(T). \\
hold(n_1(T)) & :- & time(T), holds(g(T)). \\
hold(n_2(T)) & :- & time(T), holds(neg\_g(T)). \\
hold(n_3(T)) & :- & time(T), holds(neg\_g(T)). \\
hold(n_4(T)) & :- & time(T), holds(g(T)). \\
hold(n_5(T)) & :- & time(T), holds(g(T)). \\
hold(n_6(T)) & :- & time(T), holds(g(T)). \\
hold(n_7(T, A)) & :- & time(T), action(A), holds(f(T)). \\
hold(n_8(T, A)) & :- & time(T), action(A), holds(g(T)). \\
hold(n_9(T, A)) & :- & time(T), action(A), holds(h(T)). \\
hold(n_{10}(T, A)) & :- & time(T), action(A), holds(neg\_f(T)). \\
hold(n_{11}(T, A)) & :- & time(T), action(A), holds(neg\_g(T)). \\
hold(n_{12}(T, A)) & :- & time(T), action(A), holds(neg\_h(T)).
\end{array}$$

The rules used to define the defeat of a default can be expressed as follows:

$$defated(D) \quad :- \quad default(D, L, Body), contrary(L, L1), holds(L1).$$

Observe that since we do not have specific preferences between default we can omit the second case of the *defeated* predicate.

Goal Encoding

The goal we are trying to achieve is f; this is encoded as:

$$\begin{array}{lll}
 & :- & not\ \ goal(2). \\
goal(T) & :- & time(T), holds(f(T)). \\
 & & \\
1\{occ(A, T) : action(A)\}1 & :- & time(T). \\
 & :- & action(A), time(T), occ(A, T), not\ \ holds(possible(A, T)).
\end{array}$$

Preference Encoding

The single preference we require in this example is the formula preference:

$$\neg h \prec h$$

This will be encoded as:

$$\textbf{maximize}\ \ [holds(neg\_h(2)) = 1, holds(h(2)) = 0]$$

Auxiliary

The following auxiliary predicates are employed in the encoding.

$$\begin{array}{lcl}
contrary(f(T), neg\_f(T)) & :- & time(T). \\
contrary(neg\_f(T), f(T)) & :- & time(T). \\
action(a). & & \\
action(b). & & \\
action(c). & & \\
time(1..2). & &
\end{array}$$