Formalizing Sensing Actions - a transition function based approach

Tran Cao Son and Chitta Baral
Department of Computer Science
University of Texas at El Paso
El Paso, TX 799698
{tson,chitta}@cs.utep.edu

April 29, 1998

Abstract

In presence of incomplete information about the world we need to distinguish between the state of the world and the state of the agent's knowledge about the world. In such a case the agent may need to have at its disposal sensing actions that change its state of knowledge about the world and may need to construct more general plans consisting of sensing actions and conditional statements to achieve its goal. In this paper we first develop a high level action description language that allows specification of sensing actions and their effects in its domain description and allows queries with conditional plans. We give provably correct translations of domain description in our language to axioms in first-order logic and logic programming, and relate our formulation to several earlier formulations in the literature. We then analyze the state-space of our formulation and develop several sound approximations that have much smaller state-spaces.

1 Introduction and Motivation

Unlike actions that change the world, sensing or knowledge producing actions change what the agent knows about the world. For example, the action of pushing the door ($push\_door$) – when executed in a world where the (lock of the) door is initially unlocked – will change the world so that after the action is performed the door is open. Similarly, the action of flipping the lock ($flip\_lock$) will unlock a locked door and lock an unlocked door. On the other hand the *sensing action* of checking the lock of the door ($check\_if\_locked$) will result in the agent knowing if the door is locked or not.

Sensing actions play an important role when an agent needs to plan in presence of incomplete information. Consider the case when our agent initially (i.e., in the initial situation) does not know if the door is locked or not, but knows that the door is not open and its goal is to open the door. We will assume that the only actions it can perform are the ones described earlier: $check\_if\_locked$, $flip\_lock$ and $push\_door$. It is easy to check that the agent can not just construct a classical plan – consisting of a sequence of actions – that will guaranteedly (i.e., regardless of what the real state of the world is) succeed in reaching the agent's goal. For example, neither the plan P_1 consisting of $push\_door$, nor the plan P_2 consisting of $flip\_lock; push\_door$ will guaranteedly work. That is because, it is possible that in the real world the door is initially unlocked and it is also possible that the door is initially locked. In the first case the plan P_1 will work but the plan P_2 will not. In the second case the plan P_2 will work but the plan P_1 won't. Therefore, neither P_1, nor P_2 will work in both cases.

The simple conditional plan P_3,

IF ¬ door_locked THEN push_door ELSE flip_lock;push_door

is not appropriate either. That is because, the agent not knowing whether *door_locked* is true or not can not execute this plan. A correct conditional plan, P_4, that will guaranteedly achieve the goal uses the sensing action *check_if_locked*, and is as follows:

check_if_locked;
IF ¬ door_locked THEN push_door ELSE flip_lock;push_door;

Thus sensing actions are very important for planning in presence of incomplete information. In the past, sensing actions have been formalized in [Moo79, Moo85, SL93, Haa86, LTM97] and planning in presence of incomplete information has been studied in [EHW+92, GEW96, GW96, Lev96, KOG92]. To motivate our work we now briefly review the earlier formalizations of sensing actions.

1.1 Moore's formalization

To the best of our knowledge sensing actions were first formalized by Moore in his dissertation [Moo79] and in some of his later papers, for example, [Moo85]. Moore uses possible world semantics to represent knowledge and treats the accessibility relation between worlds as a fluent when reasoning about sensing and non-sensing actions.

- He describes how the knowledge of an agent may change after executing a non-sensing action a, by defining accessibility relation between the worlds that may be reached after executing the action a.

  According to him, for any possible worlds w_1 and w_2 such that w_2 is the result of the execution of a in w_1, the worlds that are compatible with what the agent knows in w_2 are exactly the worlds that are the result of executing a in some world that is compatible with what the agent knows in w_1. This can be formally written as follows:

$$\begin{aligned}\forall w_1, w_2.(w_2 = res(a, w_1) \supset \\ \forall w_3. \quad (K(w_2, w_3) \equiv \\ \exists w_4.K(w_1, w_4) \wedge w_3 = res(a, w_4)))\end{aligned} \tag{1.1}$$

  The above formula (and the next formula) is a simplified version of Moore's original formula. Here we use situation calculus notation and assume a single agent world.

- He also describes how the knowledge of an agent may change after executing a sensing action $sense_f$, by defining accessibility relation between the worlds that may be reached after executing $sense_f$.

  Suppose $sense_f$ is an action that the agent can perform to know if f is true or not. Then for any world represented by w_1 and w_2 such that w_2 is the result of $sense_f$ happening in w_1, the world that are compatible with what the agent knows in w_2 are exactly those worlds that are the result of $sense_f$ happening in some world that is compatible with what the agent knows in w_1, and in which f has the same truth value as in w_2. This can be formally written as follows:

$$\begin{array}{lll}\forall w_1, w_2. & (w_2 = & res(sense_f, w_1) \supset \\ & \forall w_3. & ((K(w_2, w_3) \equiv \\ & & \exists w_4. K(w_1, w_4) \wedge w_3 = res(sense_f, w_4) \wedge f(w_2) \equiv f(w_3)))\end{array} \tag{1.2}$$

1.2 Scherl and Levesque's formalization

Scherl and Levesque [SL93] adapted Moore's formulation to situation calculus and proved several important results about their formulation such as: knowledge-producing actions do not affect fluents other than the knowledge fluent; and that actions that are not knowledge-producing only affect the knowledge fluent as appropriate. They also showed how regression can be applied to knowledge-producing actions.

Their slight simplification of Moore's formulation is given by the following two formulas: (Note that in their use of the relation K, which we will follow in the rest of the paper, the arguments are reversed from their normal modal logic use. I.e., $K(s's)$ is read as "the situation s' is accessible from the situation s". Also, *situation* is a term constructed by repeated application of Res to the initial situation S_0.)

$$K(s'', res(a, s)) \equiv (\exists s'. K(s', s) \wedge s'' = res(a, s')) \tag{1.3}$$

$$K(s'', res(sense_f, s)) \equiv (\exists s'. K(s', s) \wedge s'' = res(sense_f, s') \wedge f(s') \equiv f(s)) \tag{1.4}$$

1.3 Our simplification

One of our goal in this paper is to make it easy to visualize the state-space we have to deal with when searching for plans in presence of sensing actions and incomplete information. Many formulations of planning (for example, most research on decision theoretic planning) often assume the existence of a transition function defining a transition between states – a collection of fluents – due to actions, and do not necessarily depend on a logical formulation defining this function. *The questions that we would like to answer are: What is a "state" when we need to distinguish between the state of the world and the state of the knowledge of an agent? How are state transitions due to actions – both sensing and non-sensing – defined?*

To answer the first question we introduce the notion of a *k-state* (or knowledge state) which is a pair consisting of:
(i) the real state of the world (s), and
(ii) the set of states Σ, that the agent thinks it may be in.
The transition between k-states due to actions – denoted by $\Phi(a, \langle s, \Sigma \rangle)$ – can then be defined in terms of the original transition between states (defined using the function Res) in the following way:

- If a is a non-sensing action then for any k-state $\sigma = \langle s, \Sigma \rangle$, $\Phi(a, \sigma)$ is defined as the pair $\langle Res(a, s), \{s' : s' = Res(a, s'')$ for some $s'' \in \Sigma\}\rangle$.
- If $sense_f$ is a sensing action that senses the fluent f then for any k-state $\sigma = \langle s, \Sigma \rangle$ $\Phi(a, \sigma)$ is defined as the pair $\langle s, \{s' : s' \in \Sigma$ such that $f \in s$ iff $f \in s'\}\rangle$.

Consider our example in the beginning of this section. The two possible initial k-states – with explicit representation of negative fluents – for this example are:
$\sigma_1 = \langle \{locked, \neg door\_open\}, \{\{locked, \neg door\_open\}, \{\neg locked, \neg door\_open\}\}\rangle$, and
$\sigma_2 = \langle \{\neg locked, \neg door\_open\}, \{\{locked, \neg door\_open\}, \{\neg locked, \neg door\_open\}\}\rangle$. In Figure 1 we give a fragment of the state-space diagram of this example illustrating how transitions take place between one k-state to another because of actions.

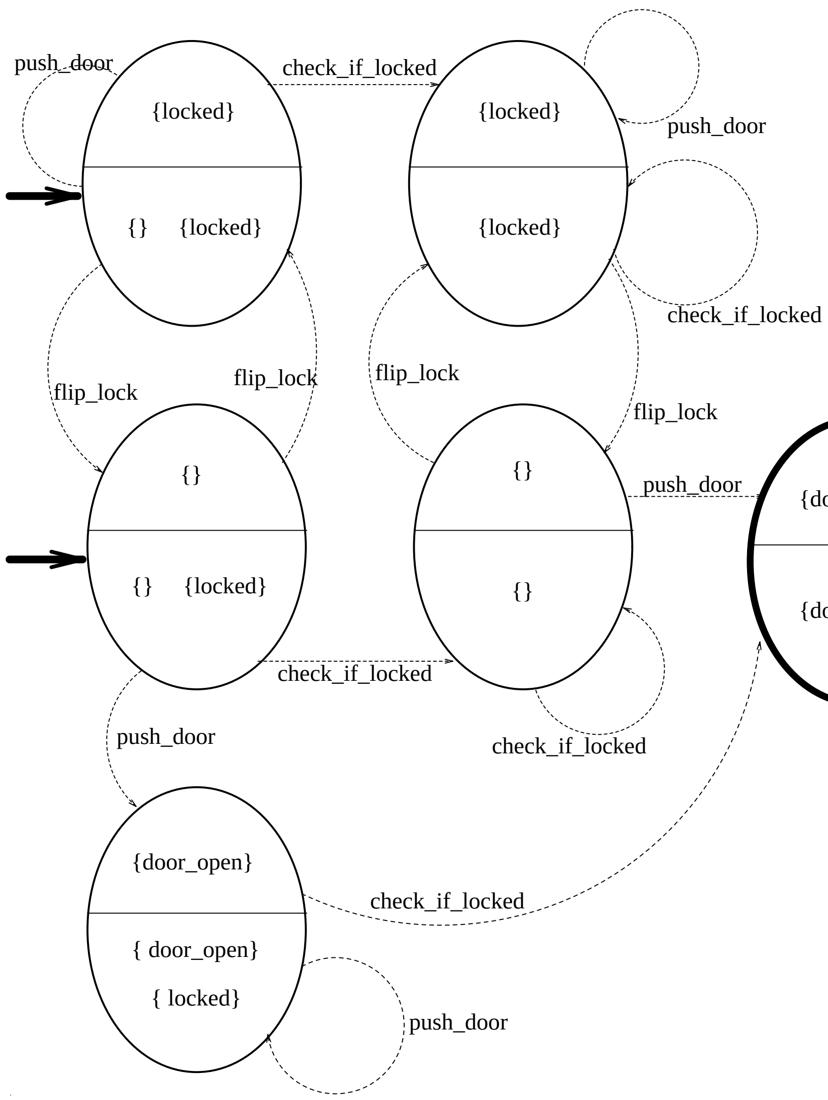

Figure 1: Transition between k-states in the door opening example.

For a logical formalization of the above we simplify Moore's and Scherl and Levesque's formulation by assuming that we only need to proceed from the K relation about the initial situation to possible future situations. The formulas 1.3 and 1.4 can then be modified as follows:

$$K(res(a, s'), res(a, s)) \equiv K(s', s) \tag{1.5}$$

$$K(res(sense_f, s'), res(sense_f, s)) \equiv (K(s', s) \wedge f(s') \equiv f(s)) \tag{1.6}$$

Using the above two formulas, successor state axioms about actions [Rei91], and information about the initial situation, we can then reason about what is known to be true in a future situation. We discuss this formulation in further detail in Section 2.3.

1.4 Our goals

Our first goal in this paper is to augment the high-level language $\mathcal{A}$ [GL92, GL93] with sensing actions – which we will call $\mathcal{A}_K$, and define its semantics using the transition functions introduced in the previous sub-section. The motivation behind doing this is the simplicity of high level languages and the fact that no knowledge about particular logics are necessary to understand the concept. But we pay the price of being less general than when the formalization is done in a standard logical language (classical logic possibly augmented with circumscription, logic programming, default logic, etc.). But then later, we give formalizations in logic, *and prove the correctness of our logical formalization* with respect to our original formalization. Thus our initial formalization using a high level language – which is simpler to follow – can play the role of a benchmark for formalizations in standard logical languages.

Our second goal, and perhaps the most important aspect of this paper, is to develop approximations of the language $\mathcal{A}_K$. The motivation behind that is the possible state space explosion in $\mathcal{A}_K$. In presence of n fluents, we will have 2^n possible states and 2^{2^n+n} possible $k-states$. We develop several approximations with much smaller state-space (3^n) but with varying complexity in computing transitions. We then show the soundness of these approximations.

Finally, we relate our formulations with earlier formulations of sensing actions – in particular with Scherl and Levesque's [SL93] formulation and Lobo et al.'s [LTM97] formulation, and show that: (i) When we translate domain descriptions in our language to Scherl and Levesque's formulation we obtain similar conclusions, and (ii) When we make certain assumptions about our knowledge about the initial state then domain descriptions in our language have the same semantics as that of the semantics defined by Lobo et al. [LTM97].

2 The language $\mathcal{A}_K$

In this section we introduce an extension of the language $\mathcal{A}$ in [GL93], $\mathcal{A}_K$, which allows reasoning about sensing actions. (Strictly speaking, $\mathcal{A}_K$ is a variation of $\mathcal{A}$ instead of an extension, as unlike in $\mathcal{A}$, we do not allow non-initial v-propositions in our domain descriptions. Moreover, our language has two components [Lif97, BGP97]: one which defines domain descriptions and another which defines queries.)

2.1 Syntax of $\mathcal{A}_K$

We begin with two disjoint nonempty sets of symbols, called *fluent names* and *action names*. A *fluent literal* is either a fluent name or a fluent name preceded by $\neg$. For a fluent f, by $\overline{\neg f}$ we mean f, and by $\overline{f}$ we mean $\neg f$.

2.1.1 Domain descriptions in $\mathcal{A}_K$

A *v-proposition* is an expression of the form

$$\textbf{initially} \quad f, \tag{2.1}$$

where f is a fluent literal. Intuitively, the above v-proposition means that the fluent literal f is initially known to be *true*. (In $\mathcal{A}$, where v-propositions describe the initial state of the world, instead of what the agent knows about the initial state of world, the above proposition has a slightly different meaning. There, the above proposition means that the fluent literal f is true in the initial state of the world.)

An *ef-proposition* is an expression of the form

$$a \ \textbf{causes} \ \ f \ \textbf{if} \ \ p_1, \ldots, p_n \tag{2.2}$$

where a is an action name, and each of $f, p_1, \ldots, p_n$ ($n \geq 0$) is a fluent literal. The set of fluent literals $\{p_1, \ldots, p_n\}$ is referred to as the *precondition* of the ef-proposition and f is referred to as the *effect* of this ef-proposition. Intuitively, this proposition conveys the meaning that f is guaranteed to be true after the execution of an action a in any state of the world where $p_1, \ldots, p_n$ are true. If $n = 0$, we will drop **if** and simply write a **causes** f.

Two ef-propositions with preconditions $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ respectively are said to be *contradictory* if they describe the effect of the same action a on complementary f's, and $\{p_1, \ldots, p_n\} \cap \{\overline{q_1}, \ldots, \overline{q_m}\} = \emptyset$.

A *k-proposition* is an expression of the form

$$a \ \textbf{determines} \ \ p \tag{2.3}$$

where a is an action name and p is a fluent. About this proposition we say that it stipulates that if a is executed in a situation, then in the resulting situation the truth value of p becomes known.

A *proposition* is a v-proposition, ef-proposition, or an k-proposition. A *domain description* is a set of propositions which does not contain contradictory ef-propositions.

Actions occurring in ef-propositions and k-propositions are called non-sensing actions and sensing actions, respectively. In this paper – to avoid distraction from the main points – we make the further assumption that the set of sensing actions and the set of non-sensing actions are disjoint. Following is an example of a domain description in our language.

Example 1 Let us consider an agent who has to *disarm* a bomb which can only be done safely – i.e., without *exploding* – if a special lock on the bomb has been switched off (*locked*); otherwise it explodes. The agent can determine if the lock is locked or not by *looking* at the lock and the agent can change the lock on the bomb (from the *locked* position to the *unlocked* position and vice-versa) by *turning* the lock. The agent initially knows that the bomb is not disarmed and it has not exploded. We can describe the above story by the following domain description.

$$\left.\begin{array}{l}
\textbf{initially} \ \ \neg disarmed \\
\textbf{initially} \ \ \neg exploded \\
disarm \ \textbf{causes} \ \ exploded \ \textbf{if} \ \ \neg locked \\
disarm \ \textbf{causes} \ \ disarmed \ \textbf{if} \ \ locked \\
disarm \ \textbf{causes} \ \ \neg exploded \ \textbf{if} \ \ locked \\
turn \ \textbf{causes} \ \ \neg locked \ \textbf{if} \ \ locked \\
turn \ \textbf{causes} \ \ locked \ \textbf{if} \ \ \neg locked \\
look \ \textbf{determines} \ \ locked
\end{array}\right\} = D_1$$

2.1.2 Queries in $\mathcal{A}_K$

As discussed in Section 1, in the presence of incomplete information and knowledge producing actions, we need to extend the notion of a plan from a sequence of actions so as to allow conditional statements. In the following definition we formalize the notion of a conditional plan.

Definition 1 (Conditional Plan)

1. An empty sequence of action, denoted by [ ], is a conditional plan.

2. If a is an action then a is a conditional plan.

3. If $c_1, \ldots, c_n$ are conditional plans and φ_j's are conjunction of fluent literals (which are mutually exclusive but not necessarily exhaustive) then the following is a conditional plan. (We refer to such a plan to as a *case plan*).

   Case
       $\varphi_1 \rightarrow c_1$
       $\ldots$
       $\varphi_n \rightarrow c_n$
   Endcase

4. If a is an action and c is a conditional plan, then $a; c$ is a conditional plan.

5. If c' is a case plan and c is a conditional plan then $c'; c$ is a conditional plan.

6. Nothing else is a conditional plan. □

Intuitively, the case plan is a case statement where the agent evaluates the various φ_i's with respect to its knowledge. If it knows that φ_i is true for some i it executes the corresponding c_i. If none of the φ_i's are true then the case plan fails and the execution of the conditional plan which contains this case plan also fails.

There are two kind of queries that we can ask our domain descriptions. They are of the form:

$$\phi \ \textbf{after} \ \ c \tag{2.4}$$

$$\textbf{Kwhether} \ \phi \ \textbf{after} \ \ c \tag{2.5}$$

where c is a conditional plan and ϕ is a fluent formula. Intuitively, the first query is about asking if a domain description entails that the fluent formula ϕ will be known to be true after executing the conditional plan c in the initial situation, and the second query is about asking if a domain description entails that the fluent formula ϕ will be known to be true or known to be false after executing the conditional plan c in the initial situation

2.2 Semantics of $\mathcal{A}_K$

In $\mathcal{A}_K$, we have two kinds of states: a world state (often referred to as a *state*) representing the state of the world, and a *knowledge state* (or a k-state), representing the state of the knowledge of the agent. As mentioned earlier, the semantics of domain descriptions in $\mathcal{A}_K$ are defined in terms of models which are pairs consisting of an initial k-state and a transition function that maps pairs of actions and k-states into k-states.

A *state* s is a set of fluents. A *knowledge state* (or *k-state*) of an agent is a pair $\langle s, \Sigma \rangle$ where s is a state and Σ is a set of states. Intuitively, the state s in a k-state $\langle s, \Sigma \rangle$ is the real state of the world whereas Σ is the set of possible states which an agent believes it might be in. We say a k-state $\sigma = \langle s, \Sigma \rangle$ is *grounded* if $s \in \Sigma$. Intuitively, grounded k-states correspond to the assumption that the real world is among the set of states that the agent believes it may be in.

Given a fluent f and a state s, we say that f holds in s (f is true in s) if $f \in s$; $\neg f$ holds in s (f is false in s) if $f \notin s$. The truth of a propositional fluent formula w.r.t. s is defined as usual. We say two states s and s' *agree on a fluent* f if $f \in s$ iff $f \in s'$. Given a k-state $\sigma = \langle s, \Sigma \rangle$, we say that a fluent f is known to be true (resp. known to be false) in $\langle s, \Sigma \rangle$ if f is true (resp. false) in every state $s' \in \Sigma$; and f is *known* in $\langle s, \Sigma \rangle$, if f is known to be true or known to be false in $\langle s, \Sigma \rangle$. Given a fluent formula φ, we say that φ is known to be true (resp. false) in a k-state $\langle s, \Sigma \rangle$ if φ is true (resp. false) in every state $s' \in \Sigma$.

For an action a and a state s, we define

$$Res(a, s) = s \cup E_a^+(s) \setminus E_a^-(s)$$

where $E_a^+(s)$ (resp. $E_a^-(s)$) is the set of fluents which becomes true (resp. false) after executing a in the state s, i.e, $E_a^+(s) = \{f \mid$ there exists an ef-proposition a **causes** f **if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ hold in $s\}$ and $E_a^-(s) = \{f \mid$ there exists an ef-proposition a **causes** $\neg f$ **if** $p_1, \ldots, p_n$ in D such that $p_1, \ldots, p_n$ hold in $s\}$. Note that since we do not allow contradictory ef-propositions in our domain description, for any state s and action a, $E_a^+(s)$ and $E_a^+(s)$ are disjoint.

Definition 2 A function Φ from actions and k-states into k-states is called a *transition function* of D if

1. for any k-state $\sigma = \langle s, \Sigma \rangle$ and non-sensing action a,

$$\Phi(a, \sigma) = \langle Res(a, s), \{s' : s' = Res(a, s'') \text{ for some } s'' \in \Sigma\} \rangle;$$

2. for any k-state $\sigma = \langle s, \Sigma \rangle$ and sensing action a whose k-propositions are
   a **determines** f_1 $\quad\ldots\quad$ a **determines** f_m

$$\Phi(a, \sigma) = \langle s, \{s' : s' \in \Sigma \text{ such that } s \text{ and } s' \text{ agree on each } f_i, (i \leq m)\} \rangle.$$

(In the first part of the above definition, there is an implicit assumption that our agent knows about each action happening in the world.) It is easy to see that for a domain description D in $\mathcal{A}_K$, the transition function Φ of D is unique. Also, notice that our definition of the transition function does not stipulate any special requirement on how the *Res* function is defined. Thus, any action description language [BG97, KL94, Tur97] with a semantics depending on a state transition function like *Res* can be extended to allow sensing actions. Thus, several of the other features of action description languages such as ramification, causality, concurrent actions, can be directly added to our framework. We do not do it here to keep our focus on the main issue of formalizing sensing actions.

Definition 3

1. A state s is called an *initial state* of a domain description D if for every value proposition of the form "**initially** p" (resp. "**initially** $\neg p$") in D, p is true (resp. false) in s.

2. A k-state $\langle s_0, \Sigma_0 \rangle$ is an *initial k-state* of D if s_0 is an initial state and Σ_0 is a set of initial states of D.

We say an initial k-state $\sigma_0 = \langle s_0, \Sigma_0 \rangle$ is *complete* if Σ_0 is the set of all initial states. Intuitively, the completeness of initial k-states express the assumption that our agent has complete knowledge about what it knows and does not know about the initial state. We will refer to this as the *complete knowledge about the initial situation assumption*[1]. Even though, we believe that this assumption should not be used indiscriminately, *since it reduces the number of initial k-states, we will use it in most of our examples.*

Definition 4 A model of a domain description D is a pair (σ_0, Φ) such that σ_0 is a grounded initial k-state of D and Φ is a transition function of D. A model (σ_0, Φ) is called *rational* if σ_0 is complete.

Since the transition function Φ as defined so far can only tell us which k-state is reached after executing *an action* in a given k-state, we need to extend the function to be able to reason – beyond action sequences – about conditional plans. When defining this extension we need to be careful about the fact that certain conditional plans can not be executed by an agent because it may lack the information to evaluate one or more conditions in the conditional plan. When this happens, we say that the resulting k-state is undefined and denote it by $\perp$. We now define the function $\hat{\Phi}$, which extends Φ.

Definition 5 1. $\hat{\Phi}([], \sigma) = \sigma$;

2. For an action a, $\hat{\Phi}(a, \sigma) = \Phi(a, \sigma)$;

3. For $c =$ Case
   $\varphi_1 \rightarrow c_1$
   $\ldots$
   $\varphi_n \rightarrow c_n$
   Endcase,

$$\hat{\Phi}(c, \sigma) = \begin{cases} \hat{\Phi}(c_i, \sigma) & \text{if } \varphi_i \text{ is known to be true in } \sigma \\ \perp & \text{if none of } \varphi_1, \ldots, \varphi_n \text{ is known to be true in } \sigma \end{cases}$$

4. For $c = a; c_1$, where a is an action and c_1 is a conditional plan, $\hat{\Phi}(c, \sigma) = \hat{\Phi}(c_1, \Phi(a, \sigma))$;

5. For $c = c_1; c_2$, where c_1 is a case plan and c_2 is a conditional plan, $\hat{\Phi}(c, \sigma) = \hat{\Phi}(c_2, \hat{\Phi}(c_1, \sigma))$;

6. $\hat{\Phi}(c, \perp) = \perp$ for every conditional plan c.

[1] Turner [Tur94] used a similar assumption called "complete initial situation assumption" according to which each models of his logic programming formulation of actions would have complete information about the initial state.

We are now ready to define the entailment relation for domains of $\mathcal{A}_K$.

Definition 6 Let D be a domain description, c be a conditional plan, and φ be a fluent formula. We say,

(i) $D \models_{\mathcal{A}_K} \varphi$ **after** c if $\hat{\Phi}(c, \sigma_0) \neq \perp$ and φ is known to be true in $\hat{\Phi}(c, \sigma_0)$ for every model (σ_0, Φ) of D;

(ii) $D \models_{\mathcal{A}_K}$ **Kwhether** φ **after** α if $\hat{\Phi}(c, \sigma_0) \neq \perp$ and φ is known to be true or known to be false in $\hat{\Phi}(\alpha, \sigma_0)$ for every model (σ_0, Φ) of D.

Rational entailment of queries w.r.t. D – denoted by $\models^r_{\mathcal{A}_K}$ – is defined similarly by only considering rational models of D. □

The following examples elucidates the above definitions.

Example 2 Let D_2 be the domain description consisting of the following propositions.

$$\left.\begin{array}{l} \textbf{initially } f \\ a \textbf{ causes } \neg f \\ sense_g \textbf{ determines } g \end{array}\right\} = D_2$$

Let $s_1 = \{f, g\}$, $s_2 = \{f\}$, $s_3 = \{g\}$, $s_4 = \emptyset$.

There are two possible complete initial k-states of D_2: $\sigma = \langle s_1, \{s_1, s_2\}\rangle$ and $\sigma' = \langle s_2, \{s_1, s_2\}\rangle$. Let Φ be the transition function of D_2. We then have:

$$\begin{array}{l} \hat{\Phi}([a], \sigma) = \langle s_3, \{s_3, s_4\}\rangle \\ \hat{\Phi}([a, sense_g], \sigma) = \hat{\Phi}([sense_g], \langle s_3, \{s_3, s_4\}\rangle) = \langle s_3, \{s_3\}\rangle \\ \\ \hat{\Phi}([a], \sigma') = \langle s_4, \{s_3, s_4\}\rangle \\ \hat{\Phi}([a, sense_g], \sigma') = \hat{\Phi}([sense_g], \langle s_4, \{s_3, s_4\}\rangle) = \langle s_4, \{s_4\}\rangle \end{array}$$

Since g is true in $\langle s_3, \{s_3\}\rangle$ and false in $\langle s_4, \{s_4\}\rangle$, we can conclude that

$D_2 \models^r_{\mathcal{A}_K}$ **Kwhether** g **after** $[a, sense_g]$.

However,

$D_2 \not\models^r_{\mathcal{A}_K}$ **Kwhether** g **after** $[a]$,

$D_2 \not\models^r_{\mathcal{A}_K}$ g **after** $[a, sense_g]$, and

$D_2 \not\models^r_{\mathcal{A}_K}$ $\neg g$ **after** $[a, sense_g]$. □

In the next example we consider conditional plans.

Example 3 Let us consider the domain description D_1 from Example 1. The states of D_1 are:

$$\begin{array}{ll} s_1 = \emptyset. & s_5 = \{disarmed\} \\ s_2 = \{locked\} & s_6 = \{disarmed, exploded\} \\ s_3 = \{exploded\} & s_7 = \{disarmed, locked\} \\ s_4 = \{locked, exploded\} & s_8 = \{disarmed, locked, exploded\} \end{array}$$

The set of initial states of D_1 is $\Sigma_0 = \{s_1, s_2\}$ and the two complete initial k-states of D_1 are $\sigma_i = \langle s_i, \Sigma_0 \rangle$ for $i = 1, 2$. Let Φ be the transition function of D_1. We now give the value of $\hat{\Phi}$ for several conditional plans and initial k-states.

$$
\begin{array}{ll}
\hat{\Phi}([look], \sigma_1) = \langle s_1, \{s_1\} \rangle & \hat{\Phi}([look], \sigma_2) = \langle s_2, \{s_2\} \rangle \\
\hat{\Phi}([look, disarm], \sigma_1) = \langle s_3, \{s_3\} \rangle & \hat{\Phi}([look, disarm], \sigma_2) = \langle s_7, \{s_7\} \rangle \\
\hat{\Phi}([look, turn], \sigma_1) = \langle s_2, \{s_2\} \rangle & \hat{\Phi}([look, turn], \sigma_2) = \langle s_1, \{s_1\} \rangle \\
\hat{\Phi}([look, turn, disarm], \sigma_1) = \langle s_7, \{s_7\} \rangle & \hat{\Phi}([look, turn, disarm], \sigma_2) = \langle s_3, \{s_3\} \rangle
\end{array}
$$

Thus we have two rational models: (σ_1, Φ) and (σ_2, Φ) and hence, we have the following:

$$D_1 \not\models^r_{\mathcal{A}_K} disarmed\ \textbf{after}\ \ [look, disarm]$$

and

$$D_1 \not\models^r_{\mathcal{A}_K} disarmed\ \textbf{after}\ \ [look, turn, disarm].$$

It is easy to see that for every sequence of actions α of D_1, $D_1 \not\models^r_{\mathcal{A}_K} disarmed$ **after** α. However, the following conditional plan helps achieving the goal.

$$
\left.
\begin{array}{l}
look; \\
\quad \text{Case} \\
\quad \neg locked \rightarrow turn \\
\qquad locked \rightarrow [] \\
\quad \text{Endcase} \\
disarm
\end{array}
\right\} = c
$$

We will now show that $D_1 \models^r_{\mathcal{A}_K} disarmed \wedge \neg exploded$ **after** c.

Consider the two rational models (σ_1, Φ) and (σ_2, Φ). It is easy to see that $\hat{\Phi}(c, \sigma_1) = \langle s_7, \{s_7\} \rangle$ and $\hat{\Phi}(c, \sigma_2) = \langle s_7, \{s_7\} \rangle$. Since *disarmed* is known to be true and *exploded* is known to be false in the k-state $\langle s_7, \{s_7\} \rangle$, we can conclude that $D_1 \models^r_{\mathcal{A}_K} disarmed \wedge \neg exploded$ **after** c. □

2.3 Translating domain descriptions to first order theories

In this section we give a translation of domain descriptions (D) in $\mathcal{A}_K$ to theories in first-order logic ($R(D)$), and then show that the translation is sound and complete w.r.t. $\mathcal{A}_K$ when answering queries in the language of $\mathcal{A}_K$. Our translation from D into $R(D)$ is inspired by the translation in Kartha [Kar93], and uses axioms and notations from [SL93] and [Rei91]. In this section we use the standard notation of having variable start with small letters and constants start with capital letters. To be consistent we use the same notation for domain descriptions.

Let us consider a domain description D. Assume that D contains $\mathbf{n}$ sensing actions $K_1, \ldots, K_{\mathbf{n}}$ with the k-propositions K_i **determines** F_i for $(1 \leq i \leq \mathbf{n})$ and $\mathbf{m}$ value-propositions **initially** G_i for $(1 \leq i \leq \mathbf{m})$. Then, the domain description D can be translated into a many-sorted theory $R(D)$ as follows.

Objects of $R(D)$ are of the sorts: *action, fluent*, and *situation*. The vocabulary (signature) of $R(D)$ consists of the following:

- a constant S_0 of type situation;
- constants A of type "action" which correspond to different actions from D (one constant for each action);

- constants F of type "fluent" which correspond to different fluents from D (one constant for each fluent);
- a function symbol do of the type $\langle action, situation \rightarrow situation\rangle$;
- a predicate symbol $Holds$ of the type $\langle fluent, situation\rangle$;
- a predicate symbol K of the type $\langle situation, situation\rangle$;

We will need the following notation. For each fluent name F of D, let

$A_F^+ = \{A \textbf{ causes } F \textbf{ if } \varrho : A \textbf{ causes } F \textbf{ if } \varrho \text{ in } D\}$, and

$A_F^- = \{A \textbf{ causes } \neg F \textbf{ if } \varrho : A \textbf{ causes } \neg F \textbf{ if } \varrho \text{ in } D\}$.

We define[2]:

$\gamma_F^+(a,s) \stackrel{def}{\equiv} \bigvee_{A \textbf{ causes } F \textbf{ if } \varrho \in A_F^+}(a = A \wedge \varrho(s))$, and

$\gamma_F^-(a,s) \stackrel{def}{\equiv} \bigvee_{A \textbf{ causes} \neg F \textbf{ if } \varrho \in A_F^-}(a = A \wedge \varrho(s))$.

The axioms of $R(D)$ are described below.

1. The successor state axioms – using Reiter's formulation in [Rei91] – for an ordinary fluent F is given by

$$Holds(F, do(a,s)) \equiv \gamma_F^+(a,s) \vee (Holds(F,s) \wedge \neg\gamma_F^-(a,s)) \tag{2.6}$$

$$Holds(\neg F, do(a,s)) \equiv \gamma_F^-(a,s) \vee (Holds(\neg F,s) \wedge \neg\gamma_F^+(a,s)) \tag{2.7}$$

2. The successor state axiom for K (borrowed from [SL93]) is given by:

$$\begin{gathered}[K(s'', do(x,s)) \equiv \\ \exists s' \ (K(s',s) \wedge (s'' = do(x,s')))\wedge \\ ((\textstyle\bigwedge_{j=1}^{\mathbf{n}} x \neq K_j) \vee (\bigvee_{j=1}^{\mathbf{n}}(x = K_j \wedge Holds(F_j,s) \equiv Holds(F_j,s'))))]\end{gathered} \tag{2.8}$$

3. For $i = 1, \ldots, \mathbf{m}$, $R(D)$ contains

$$Holds(G_i, S_0) \tag{2.9}$$

4. The following axioms are for the accessibility relation in the initial situation:

$$K(s, S_0) \supset \bigwedge_{i=1}^{\mathbf{m}} Holds(G_i, s). \tag{2.10}$$

and

$$K(S_0, S_0). \tag{2.11}$$

[2]For a conjunction of literals $\varrho = F_1, \ldots, F_k$, we write $Holds(\varrho, s)$ to denote $\bigwedge_{i=1}^{k} Holds(F_i, s)$.

5. The domain closure assumption (DCA) for fluents:

$$\bigvee_{F \in \mathbf{F}} f = F.$$

6. The domain closure assumption (DCA) for actions:

$$\bigvee_{A \in \mathbf{A}} a = A.$$

7. The unique name assumption (UNA) for fluents:

$$\bigwedge_{F_1, F_2 \in \mathbf{F} \;\; F_1, F_2 \text{ distinct}} F_1 \neq F_2.$$

8. The unique name assumption (UNA) for actions:

$$\bigwedge_{A_1, A_2 \in \mathbf{A} \;\; A_1, A_2 \text{ distinct}} A_1 \neq A_2.$$

We now relate the entailment in D and the entailment in $R(D)$, for queries regarding fluent values after a sequence of actions. In it we use $Knows(\varphi, S)$ as a shorthand for the formula: $\forall s'(K(s', S) \supset Holds(\varphi, s'))$, where $Holds(\varphi, s')$ is a shorthand for a corresponding formula of $Holds$ with only fluents as its first argument.

Proposition 1 Let D be a domain description, φ be a fluent formula, and α be a sequence of actions of D. Then,

$$D \models_{\mathcal{A}_K} \varphi \ \mathbf{after} \ \alpha \quad \text{iff} \quad R(D) \models Knows(\varphi, do(\alpha, S_0)).$$

Proof. In Appendix A. □

Our next step is to relate D and $R(D)$ for queries with conditional plans. For that we introduce a three-sorted predicate $Apply(c, s, s')$, whose intuitive meaning is that the conditional plan c executed in situation s takes us to the situation s'. (The definition of 'apply' is similar to the formula 'Rdo' in [Lev96].) We will represent a case plan as a list of pairs of conditions and conditional plans using three constructor functions: one that constructs a list, another that constructs a pair and the third one is the function 'case' that constructs a case plan. Any conditional plan can be represented as a list of actions and case plans. For example, the conditional plan c in the Example 3 is represented by $[look,\ case([([\neg locked], [turn]), ([locked], [])]),\ disarm]$. We now define $Apply$ as a nested abnormality theory (NAT) [Lif95] block.

$B_{\text{Apply}} =$
$\{\min \quad Apply \ :$
$\quad Apply([], s, s)$
$\quad Apply(\alpha, do(a, s), s') \supset Apply([a|\alpha], s, s')$
$\quad Apply([case([])|r], s, \perp)$
$\quad Apply(r, \perp, \perp)$
$\quad Knows(\varphi, s) \wedge Apply(r, s, s') \wedge Apply(r'', s', s'') \supset Apply([case([(\varphi, r)|r'])|r''], s, s'')$
$\quad \neg Knows(\varphi, s) \wedge Apply([case(r')|r''], s, s')) \supset Apply([case([(\varphi, r)|r'])|r''], s, s')$
$\}$

To relate D and $R(D)$ w.r.t. queries with conditional plans, we now use $Knows(\varphi, S)$ as a shorthand for $S \neq \perp \wedge \forall s'(K(s', S) \supset Holds(\varphi, s'))$.

Proposition 2 Let D be a domain description and $R(D)$ be the corresponding first order theory. Let c be a conditional plan and φ be a fluent formula. Then,

$$D \models \varphi \ \textbf{after} \ c \quad \text{iff} \quad R(D) \cup B_{\text{Apply}} \models Apply(c, S_0, s) \wedge Knows(\varphi, s).$$

Proof. In Appendix A. $\Box$

We would like to point out that the above proposition also holds for a slightly different translation $R_1(D)$, where we use the following simpler successor state axiom – based on the formulas (1.5) and (1.6) of Section 1.3 – instead of the successor state axiom (2.8):

$$\begin{gathered}[K(do(x,s'), do(x,s)) \equiv \\ (K(s',s) \wedge ((\bigwedge_{j=1}^{\mathbf{n}} x \neq K_j) \vee (\bigvee_{j=1}^{\mathbf{n}} (x = K_j \wedge Holds(F_j, s) \equiv Holds(F_j, s')))))]\end{gathered} \tag{2.12}$$

2.4 Translating domain descriptions to disjunctive logic programs

Logic programming has been one of the standard logical languages that has been used in formalizing reasoning about actions [BG94, AB91] and has also been used in translating domain descriptions in high-level action description languages [GL93, BG97, Bar95, Tur97, DMB92, Dun93, Sha97]. Besides being a widely studied knowledge representation language with regular international conferences (for example: International conference on logic programming; International logic programming symposium, and International conference on logic programming and non-monotonic reasoning), the programming language PROLOG is based on logic programming. Disjunctive logic programming is one of the most expressive declarative logic programming language. Although PROLOG is not quite appropriate for disjunctive logic programs, other systems have been developed (such as the XSB system [CSW95]) that allow disjunctive logic programs.

We now present a translation from any domain description (D) in $\mathcal{A}_K$ into a disjunctive logic program $\pi(D)$ and show that the translation (using answer-set semantics [GL91]) is sound and complete w.r.t. the semantics of $\mathcal{A}_K$. *We have executed translations of several domain descriptions in the XSB system to reason about sensing actions and conditional plans.* Here we follow the notation of logic programming and have variables start with capital letters and constants start with lower case letters. (We also revert back to our original usage of lower case letters for constants in domain descriptions of $\mathcal{A}_K$.)

The translation $\pi(D)$ of a domain description D, uses variables of three sorts: *situation* variables $S, S', \ldots$, *fluent* variables $F, F', \ldots$, and *action* variables $A, A', \ldots$. Lower case letters are used to denote constants of the same sort as its upper case counterpart. In our translation, for a fluent f, by $Holds(f,s)$ we denote $holds(f,s)$ and by $Holds(\neg f, s)$ we denote $\neg holds(f,s)$. Also, for a fluent f, by $\bar{f}$ we denote $\neg f$ and by $\overline{\neg f}$ we denote f.

In our translation we have the situation constant s_0 that denotes the initial situation. But besides s_0, in presence of n fluents we have $s_1, \ldots, s_{2^n}$ other situation constants which are reference situation constants. (This differs from the standard formalization of actions in logic programming where s_0 is the only situation constant in the language.) Situations are terms constructed using the function symbol *Res*, actions and situation constants. Situations that are built on a situation constant other than s_0 are referred to as reference situations and are used for defining accessibility.

Intuitively, $holds(F,S)$ (resp. $\neg holds(F,S)$) means that the fluent F is true (resp. false) in the situation S. Conditional plans are represented as lists as in the previous section. The agree relation between situations is captured in the translation by the relation $agree(F,S,S')$ which means that the two situations S and S' agree on the fluent F.

Given a domain description D, $\pi(D)$ consists of the following rules:

- **Domain dependent rules**:

  The v-propositions, ef-propositions and k-propositions of the form (2.1), (2.2) and (2.3), respectively, are represented by the following rules:

$$initially(f) \tag{$\pi.1$}$$

$$causes(a, f, [p_1, \dots, p_n]) \tag{$\pi.2$}$$

$$determines(a, f) \tag{$\pi.3$}$$

- **Domain independent rules**:

  - **The inertial rule**

$$\begin{aligned} Holds(F, res(A,S)) \leftarrow\ & Holds(F,S), \\ & not\ ab(F,A,S). \end{aligned} \tag{$\pi.4$}$$

  - **Rules describing effects of actions:**

$$holds\_list([], S). \tag{$\pi.5$}$$

$$\begin{aligned} holds\_list([H|T], S) \leftarrow\ & Holds(H,S), \\ & holds\_list(T,S). \end{aligned} \tag{$\pi.6$}$$

$$\begin{aligned} Holds(F, res(A,S)) \leftarrow\ & causes(A,F,L), \\ & holds\_list(L,S). \end{aligned} \tag{$\pi.7$}$$

$$\begin{aligned} ab(\bar{F}, A, S) \leftarrow\ & causes(A,F,L), \\ & holds\_list(L,S). \end{aligned} \tag{$\pi.8$}$$

$$\begin{aligned} accessible(res(A,S_2), res(A,S_1)) \leftarrow\ & causes(A,F,L), \\ & accessible(S_2,S_1). \end{aligned} \tag{$\pi.9$}$$

$$\neg accessible(res(A,S_2), res(A,S_1)) \leftarrow determines(A,F), \tag{$\pi.10$}$$

$$
\begin{aligned}
& & & accessible(S_2, S_1), \\
& & & \neg agree(F, S_1, S_2).
\end{aligned}
$$

$$
\begin{aligned}
accessible(res(A, S_2), res(A, S_1)) & \leftarrow & & determines(A, F), \\
& & & accessible(S_2, S_1), \\
& & & not\ \neg accessible(res(A, S_2), res(A, S_1))
\end{aligned} \qquad (\pi.11)
$$

Rules in this group describe the effects of actions. The first two rules define $holds\_list$ in terms of $holds$, i.e., they define when a list of fluent literals hold in a situation. The third and the fourth rule define the effect of non-sensing actions on fluent values. The fifth rule defines the effect of non-sensing actions on the accessibility relation. The last two rules define the effect of sensing actions on the accessibility relation.

– **Rules about the initial states:**

$$holds(F, s_0) \leftarrow initially(F). \qquad (\pi.12)$$

$$\neg holds(F, s_0) \leftarrow initially(\neg F). \qquad (\pi.13)$$

$$holds(F, s_0) \vee \neg holds(F, s_0). \qquad (\pi.14)$$

$$\neg accessible(S, s_0) \leftarrow state(S), initially(F), Holds(\bar{F}, S). \qquad (\pi.15)$$

$$accessible(S, s_0) \leftarrow state(S), not\ \neg accessible(S, s_0). \qquad (\pi.16)$$

Rules in this group describe complete initial k-states of a domain description, i.e., the initial state s_0 and the accessibility relation at s_0. Rules (π.12) and (π.13) say that if F is known to be true (false) initially then $holds(F, s_0)$ ($\neg holds(F, s_0)$) holds. Rule (π.14) indicates that for every fluent F, either $Holds(F, s_0)$ or $Holds(\bar{F}, s_0)$ holds. (Turner in [Tur94] calls this 'complete initial situation axiom'.) Rules (π.15) and (π.16) *define the accessibility relation such that the initial k-state is complete.*

– **Rules for encoding the** $\neg agree$ **relationship:**

$$\neg agree(F, S_1, S_2) \leftarrow Holds(F, S_1), Holds(\bar{F}, S_2). \qquad (\pi.17)$$

$$\neg agree(F, S_2, S_1) \leftarrow Holds(F, S_1), Holds(\bar{F}, S_2). \qquad (\pi.18)$$

These rules define when the two states S_1 and S_2 do not agree on a fluent F.

– **Rules for answering queries:** Rules in this group are designated for answering queries of the form (2.4) and (2.5).

$$apply([], S, S). \tag{π.19}$$

$$apply([A|T], S, S1) \leftarrow action(A), apply(T, res(A, S), S1). \tag{π.20}$$

$$\neg knows(F, S_1) \leftarrow accessible(S_2, S_1), Holds(\bar{F}, S_2). \tag{π.21}$$

$$knows(F, S) \leftarrow not\ \neg knows(F, S). \tag{π.22}$$

$$kwhether(F, S) \leftarrow knows(F, S). \tag{π.23}$$

$$kwhether(F, S) \leftarrow knows(\bar{F}, S). \tag{π.24}$$

$$knows\_after(F, C, S) \leftarrow apply(C, S, S1), S1 \neq \perp, knows(F, S1). \tag{π.25}$$

$$knows\_list([], S). \tag{π.26}$$

$$knows\_list([H|T], S) \leftarrow knows(H, S), knows\_list(T, S). \tag{π.27}$$

$$apply(X, \perp, \perp). \tag{π.28}$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow knows\_list(C, S), apply(P, S, S1), apply(L, S1, X). \tag{π.29}$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow not\ knows\_list(C, S), apply([case(T)|L], S, X). \tag{π.30}$$

$$apply([case([])|L], S, \perp). \tag{π.31}$$

The rules (π.19), (π.20), and (π.28)-(π.31) define the predicate apply such that its third argument is the situation obtained by applying the conditional plan in its first argument to the situation in its second argument. The rules (π.21) and (π.22) define when a fluent literal is known to be true in a situation and the rules (π.23) and (π.24) define when the truth value of a fluent literal is known in a situation. The rules (π.26)-(π.27) define when a list of fluent literals are known in a situation. Finally the rule (π.25) defines when a fluent literal f is known to be true after executing a conditional plan c in a situation s.

- **Language dependent rules:**

– **Enumerating situation constants corresponding to all possible states**
If the fluents in the language are $f_1, \ldots, f_n$, then we enumerate all possible states by situation constants $s_1, \ldots, s_{2^n}$. For example, a domain with 2 fluents f, g has four situation constants that map to all possible states. s_1 maps to $\{f, g\}$, s_2 maps to $\{f\}$, s_3 maps to $\{g\}$, and s_4 maps to $\emptyset$. (Often we will abuse notation and just write $s_1 = \{f, g\}$.) These four situation constants and their mapping are enumerated by the following rules:

$$\begin{array}{rrrr} state(s_1). & state(s_2). & state(s_3). & state(s_4). \\ holds(f, s_1). & holds(g, s_1). & holds(f, s_2). & \neg holds(g, s_2). \\ \neg holds(f, s_3). & holds(g, s_3). & \neg holds(f, s_4). & \neg holds(g, s_4). \end{array}$$

– **Listing all actions.**
For every action a, in the language we have:

$$action(a).$$

In the next examples, we present $\pi(D_2)$ to illustrate the translation of $\mathcal{A}_K$-domains into logic programs.

Example 4 Consider the domain description D_2 from Example 2. The program $\pi(D_2)$ will consist of the domain independent rules (π.4)-(π.31), the domain dependent rules, and the rules for enumerating the possible states of D_2. The domain dependent rules of $\pi(D_2)$ are as follows.

$$initially(f).$$

$$\begin{array}{r} causes(a, f, []). \\ determines(sense_g, g). \end{array}$$

Rules enumerating the set of possible states for D_2 are:

$$\begin{array}{rrrr} holds(f, s_1). & holds(g, s_1). & holds(f, s_2). & \neg holds(g, s_2). \\ \neg holds(f, s_3). & holds(g, s_3). & \neg holds(f, s_4). & \neg holds(g, s_4). \end{array}$$

It is easy to check that $\pi(D_2)$ has two answer sets. Both answer sets contain $accessible(s_1, s_0)$, and $accessible(s_2, s_0)$. One answer set, which we will refer as A_1, contains $holds(g, s_0)$ and the other, which we will refer as A_2, contains $\neg holds(g, s_0)$. Furthermore, A_1 contains $knows(g, res(sense_g, s_0))$ and A_2 contains $knows(\neg g, res(sense_g, s_0))$. This implies that

$\pi(D_2) \models kwhether(g, res(sense_g, s_0))$,

$\pi(D_2) \not\models knows(g, res(sense_g, s_0))$, and

$\pi(D_2) \not\models knows(\neg g, res(sense_g, s_0))$. □

The following proposition relates the rational entailment of D and entailment in $\pi(D)$ w.r.t. queries.

Proposition 3 (Soundness and Completeness) Let D be a domain description and f be a fluent in D. Then, for any conditional plan c,

$$\pi(D) \models knows\_after(f, c, s_0) \quad \text{iff} \quad D \models^r_{\mathcal{A}_K} f \textbf{ after } c.$$

Proof: In Appendix C. □

2.5 State-space analysis

In this section we analyze the size of the state-space, when reasoning in $\mathcal{A}_K$.

• When we have n fluents, we will have 2^n different states. Recall that a k-state is a pair $\langle s, \Sigma \rangle$, where s is a state, and Σ is a set of states. Thus Σ can take any value among 2^{2^n} possible values and s can take any value among 2^n values. Therefore, the total number of k-states will be $2^n \times (\binom{2^n}{0} + \binom{2^n}{1} + \binom{2^n}{2} + \ldots + \binom{2^n}{2^n}) = 2^n \times 2^{2^n} = 2^{2^n+n}$.

• If we consider only grounded k-states, the total number of grounded k-states will be

$$\binom{2^n}{1} + 2 \times \binom{2^n}{2} + \ldots + 2^n \times \binom{2^n}{2^n} = 2^n \times 2^{2^n-1} = 2^{2^n+n-1}$$

where $r \times \binom{2^n}{r}$ corresponds to number of k-states with the Σ part having r number of states.

• Now suppose out of the n fluents, in the initial situation we do not know the truth value of p ($p \leq n$) fluents. I.e., we know the truth value of $n - p$ fluents. Then in all initial k-states $\langle s, \Sigma \rangle$, the size of Σ will be less than 2^p. Since we do not have any knowledge loosing actions, and the way our transition function is defined, any k-state that can be reached by executing a sequence of actions in the initial k-state will also have the size of its Σ less than 2^p. Taking this into account the size of the reachable (from the initial k-states) state-space will be:

$$\binom{2^n}{1} + 2 \times \binom{2^n}{2} + \ldots + 2^p \times \binom{2^n}{2^p}$$

which is larger than 2^{2^p}.

• If we consider the formulations in [Moo85, SL93] the 'states' will be Kripke models. In that case for n fluents, we will have at least 2^n different possible worlds, and the accessibility relation will be a subset of $2^n \times 2^n = 2^{2n}$ elements. Thus the total number of different Kripke-models will be $2^n \times 2^{2^{2n}} = 2^{2^{2n}+n}$.

The tremendously large size of the state-space for $\mathcal{A}_K$ and also for the formulations in [SL93, LTM97], necessitates search for (provenly sound) approximations that have a more manageable state-space. This is our focus in the next section.

3 Approximating $\mathcal{A}_K$

In this section we define several approximations of the semantics of $\mathcal{A}_K$. In our approximations we will use 3-valued states, which we will call *a-states* (or approximate states), to represent the state of knowledge of an agent. An a-state will be normally represented by a pair $\langle T, F \rangle$, where T and F are disjoint sets of fluent names. Intuitively, T (resp. F) is the set of fluents which are true (resp. false) in the state $\langle T, F \rangle$. An a-state $\langle T, F \rangle$ is said to be *complete* if $T \cup F$ is the set of all the fluents in the domain. Often we will abuse notation to represent a complete a-state $\langle T, F \rangle$, by just T. Let $\sigma_1 = \langle T_1, F_1 \rangle$ and $\sigma_2 = \langle T_2, F_2 \rangle$ be two a-states. We say the a-state $\langle T_1, F_1 \rangle$ extends the a-state $\langle T_2, F_2 \rangle$, denoted by $\sigma_2 \preceq \sigma_1$, if $T_2 \subseteq T_1$ and $F_2 \subseteq F_1$. $\sigma_1 \cap \sigma_2$ will denote $\langle T_1 \cap T_2, F_1 \cap F_2 \rangle$ and by $\sigma_1 \setminus \sigma_2$ we denote the set $(T_1 \setminus T_2) \cup (F_1 \setminus F_2)$, and for a set of fluents X we write $X \setminus \langle T, F \rangle$ to denote $X \setminus (T \cup F)$.

Given a fluent name f and an a-state $\sigma = \langle T, F \rangle$, we say that f is *true* (resp. *false*) in σ if $f \in T$ (resp. $f \in F$); and f is *known* (resp. *unknown*) in σ if $f \in T \cup F$ (resp. $f \notin T \cup F$). A positive (resp. negative) fluent literal f is said to hold in $\langle T, F \rangle$ if $f \in T$ (resp. $\overline{f} \in F$).

We are now ready to define several approximations for $\mathcal{A}_K$. The difference between the approximations is based on how much case analysis is done to reason about actions when the agent has incomplete knowledge about the world. We start with the 0-Approximation where no case analysis is done.

3.1 0-Approximation

Let D be a domain description, $\langle T, F\rangle$ be an a-state, and f be a fluent name in D. f (resp. $\neg f$) is said to *possibly hold* in $\langle T, F\rangle$ if $f \notin F$ (resp. $f \notin T$). A set of fluents $\{f_1, \ldots, f_n\}$ is said to *possibly hold* in $\langle T, F\rangle$ if for all i, f_i possibly holds in $\langle T, F\rangle$. We now introduce several notations.

- $e_a^+(\langle T, F\rangle)$ denotes the set $\{f \mid f$ is a fluent name and there exists an ef-proposition in our domain description whose action is a, effect is f, and whose preconditions hold in $\langle T, F\rangle\}$.
- $e_a^-(\langle T, F\rangle)$ denotes the set $\{f \mid f$ is a fluent name and there exists an ef-proposition in our domain description whose action is a, effect is $\neg f$, and whose preconditions hold in $\langle T, F\rangle\}$.
- $F_a^+(\langle T, F\rangle)$ denotes the set $\{f \mid f$ is a fluent name and there exists an ef-proposition in our domain description whose action is a, effect is f, and whose preconditions possibly hold in $\langle T, F\rangle\}$.
- $F_a^-(\langle T, F\rangle)$ denotes the set $\{f \mid f$ is a fluent name and there exists an ef-proposition in our domain description whose action is a, effect is $\neg f$, and whose preconditions possibly hold in $\langle T, F\rangle\}$.
- $K(a, \langle T, F\rangle)$ denotes the set $\{f \mid f$ is a fluent name and there exists a k-proposition a **determines** f in our domain description. $\}$

Intuitively, $e_a^+(\langle T, F\rangle)$ (resp. $e_a^-(\langle T, F\rangle)$) is the set of fluents that *must be true* (resp. *false*) after executing a in the a-state $\langle T, F\rangle$; $F_a^+(\langle T, F\rangle)$ (resp. $F_a^-(\langle T, F\rangle)$) is the set of fluents that *may be true* (resp. *false*) after executing a in the a-state $\langle T, F\rangle$; and $K(a, \langle T, F\rangle)$ is the set of fluents which become known after executing the action a in the a-state $\langle T, F\rangle$.

We define the result function of D as follows.

$$Res(a, \langle T, F\rangle) = \langle T \cup e_a^+ \setminus F_a^-, F \cup e_a^- \setminus F_a^+\rangle.$$

We illustrate these definitions in the next example.

Example 5 For the domain description D_1 from Example 1, the initial a-state is $\sigma_0 = \langle \emptyset, \{disarmed, exploded\}\rangle$.

Since neither $locked$ nor $\neg locked$ holds in σ_0, we have that

$$e^+_{disarm}(\sigma_0) = \emptyset \qquad e^-_{disarm}(\sigma_0) = \emptyset$$

$$e^+_{turn}(\sigma_0) = \emptyset \qquad e^-_{turn}(\sigma_0) = \emptyset$$

Since $locked$ and $\neg locked$ possibly hold in σ_0, we have that

$$F^+_{disarm}(\sigma_0) = \{exploded, disarmed\} \qquad F^-_{disarm}(\sigma_0) = \{exploded\}$$

$$F^+_{turn}(\sigma_0) = \{locked\} \qquad F^-_{turn}(\sigma_0) = \{locked\}$$

Hence, $K(look, \langle T,F\rangle) = \{locked\}$ for every a-state $\langle T,F\rangle$ of D.

$$Res(disarm, \sigma_0) = \langle \emptyset, \{disarmed\}\rangle$$

$$Res(turn, \sigma_0) = \langle \emptyset, \{disarmed, exploded\}\rangle$$

□

Definition 7 Given a domain description D, the 0-transition function Φ of D is defined by

$$\Phi(a,\sigma) = \{\sigma' \mid Res(a,\sigma) \preceq \sigma', \sigma' \setminus Res(a,\sigma) = K(a,\sigma) \setminus Res(a,\sigma)\}.$$

□

Note the difference between the definition of 0-transition function above and the definition of transition function in $\mathcal{A}$, where we do not have any sensing actions. In $\mathcal{A}$, the transition function is defined in terms of the Res function. Here, $Res(a,\sigma)$, only specifies what is known to be true and what is known to be false in all a-states that may be reached by executing a in the a-state σ. The 0-transition function Φ takes into account $Res(a,\sigma)$ and $K(a,\sigma)$ to define the set of a-states that may be reached by executing a in σ.

Let Φ be a 0-transition function of D. In what follows, we define the 0-extended transition function $\hat{\Phi}$ from conditional plans and a-states into set of a-states. As before, executing a conditional plan might result in an undefined a-state, denoted by $\perp$.

Definition 8

1. $\hat{\Phi}([], \sigma) = \{\sigma\}$;
2. $\hat{\Phi}(a,\sigma) = \Phi(a,\sigma)$;
3. For a case plan

$$\begin{array}{l} c = \text{Case} \\ \quad \varphi_1 \rightarrow p_1 \\ \quad \vdots \\ \quad \varphi_n \rightarrow p_n \\ \text{Endcase} \end{array}$$

$$\hat{\Phi}(c,\sigma) = \begin{cases} \hat{\Phi}(p_j,\sigma) \text{ if } \varphi_j \text{ holds in } \sigma \\ \\ \{\perp\} \quad \text{if none of } \varphi_1,\ldots,\varphi_n \text{ holds in } \sigma \end{cases}$$

4. For an action a and a conditional plan c , $\hat{\Phi}([a;c],\sigma) = \hat{\Phi}(c, (\Phi(a,\sigma))$;
5. For a case plan c_1 and a conditional plan c_2, $\hat{\Phi}([c_1;c_2],\sigma) = \bigcup_{\sigma' \in \hat{\Phi}(c_1,\sigma)} \hat{\Phi}(c_2,\sigma')$;
6. $\hat{\Phi}(c,\perp) = \{\perp\}$

Definition 9 Given a domain description D, we say a structure (σ_0, Φ) is a 0-model of D if

(i) for any fluent literal f, f holds in σ_0 iff **initially** $f \in D$,

(ii) Φ is a transition function of D. $\square$

It is easy to see that for a domain description D in $\mathcal{A}_K$, the transition function Φ of D is unique. In the next definition, we define our first approximate entailment relation, the 0-entailment ($\models_0$), based on the 0-model.

Definition 10 Let D be a domain description, φ be a formula of fluents of D, and c be a conditional plan in D. We say $D \models_0 \varphi$ **after** c if for every 0-model (σ_0, Φ) of D,

(i) $\bot \not\in \hat{\Phi}(\alpha, \sigma_0)$; and

(ii) φ holds in every a-state belonging to $\hat{\Phi}(\alpha, \sigma_0)$.

$\square$

Example 6 For the domain description D_1 we have that

$$\begin{array}{lcl}
\Phi(disarm, \sigma_0) & = & \{\langle \emptyset, \{disarmed\}\rangle\} \\
\Phi(turn, \sigma_0) & = & \{\langle \emptyset, \{disarmed, exploded\}\rangle\} \\
\Phi(look, \sigma_0) & = & \{\langle \{locked\}, \{disarmed, exploded\}\rangle, \\
 & & \langle \emptyset, \{locked, disarmed, exploded\}\rangle\}
\end{array}$$

It is easy to see that $D_1 \models_0$ **Kwhether** $locked$ **after** $look$ but
$D_1 \not\models_0 locked$ **after** $look$ and $D_1 \not\models_0 \neg locked$ **after** $look$. $\square$

In the next example we show that the conditional plan for disarming the bomb in Example 3 can also be analyzed using the 0-Approximation.

Example 7 Let us consider again the domain D_1 and the conditional plan discussed in Example 3.

$$\left.\begin{array}{l}
look; \\
\left.\begin{array}{l}
\quad \text{Case} \\
\quad \neg locked \rightarrow turn \\
\qquad locked \rightarrow [\,] \\
\quad \text{Endcase}
\end{array}\right\} = c_1 \\
disarm
\end{array}\right\} = c$$

To prove that $D_1 \models_0 disarmed \wedge \neg exploded$ **after** c we compute $\hat{\Phi}(c, \sigma_0)$ as follows.

Since $K(look, \langle T, F\rangle) = \{locked\}$ for any a-state $\langle T, F\rangle$, we have that

a. $\Phi(look, \sigma_0) = \{\langle \{locked\}, \{disarmed, exploded\}\rangle, \langle \emptyset, \{disarmed, exploded, locked\}\rangle\}$.

This implies that

b. $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c_1; disarm, \sigma_1) \cup \hat{\Phi}(c_1; disarm, \sigma_2)$

where $\sigma_1 = \langle \{locked\}, \{disarmed, exploded\}\rangle$ and $\sigma_2 = \langle \emptyset, \{disarmed, exploded, locked\}\rangle$.

It is easy to check that

c. $\hat{\Phi}(c_1; disarm, \sigma_1) = \hat{\Phi}(disarm, \sigma_1)$

and

d. $\hat{\Phi}(disarm, \sigma_1) = \{\langle\{disarmed, locked\}, \{exploded\}\rangle\}$.

Furthermore,

e. $\hat{\Phi}(c_1; disarm, \sigma_2) = \hat{\Phi}(disarm, \hat{\Phi}(turn, \sigma_2))$

and

f. $\hat{\Phi}(turn, \sigma_2) = \{\langle\{locked\}, \{disarmed, exploded\}\rangle\} = \{\sigma_1\}$.

From (d)-(f) we conclude that

g. $\hat{\Phi}(c_1; disarm, \sigma_2) = \{\langle\{disarmed, locked\}, \{exploded\}\rangle\}$.

It follows from (d) and (g) that $\hat{\Phi}(c, \sigma_0) = \{\langle\{disarmed, locked\}, \{exploded\}\rangle\}$ which implies that $D_1 \models_0 disarmed \wedge \neg exploded$ **after** c. □

Although 0-Approximation can correctly analyze the above example, it has weaknesses and it can not entail many queries entailed by the $\mathcal{A}_K$ semantics. The following example illustrates this.

Example 8 Let us consider the domain D_3 with the following causal rules;

$$\left.\begin{array}{l} a \textbf{ causes } f \textbf{ if } g \\ a \textbf{ causes } f \textbf{ if } \neg g \end{array}\right\} = D_3$$

The initial a-state of D_3 is $\sigma_0 = \langle\emptyset, \emptyset\rangle$. Intuitively, we would expect that f **after** a is entailed by D_3 and this entailment holds for $\models_{\mathcal{A}_K}$. However, it is easy to check that $D_3 \not\models_0 f$ **after** a. □

In the above example we can intuitively conclude that f should be true after executing a in the initial situation, by doing case analysis. I.e., we analyze that in the initial situation g could be either true or false, and in both cases we can conclude that f will be true after executing a. The reasoning mechanism in the 0-Approximation lacks any such case analysis. In the next section we introduce the notion of 1-Approximation that does some case analysis and is able to make the intuitive conclusion in the above example.

3.2 1-Approximation

The 1-Approximation improves on 0-Approximation by defining a new result function which given an incomplete state σ and an action a, considers all complete extensions of σ, and applies a to these extension and then considers what is true and what is false in all the resulting states. Such a transition function does intuitive reasoning w.r.t. the Example 8. We now formally define the new result function. For an a-state σ, let $Comp(\sigma)$ be the set of all the complete a-states that extend σ. The result function, $\mathcal{R}es$, which maps a pair of action and a-state (a, σ) into an a-state $\mathcal{R}es(a, \sigma)$ is defined as follows.

$$\mathcal{R}es(a, \sigma) = \bigcap_{\sigma' \in Comp(\sigma)} Res(a, \sigma').$$

The 1-transition function is defined next.

Definition 11 Let D be a domain description. A function Φ which maps a pair of action a and a-state σ into a set of a-states, denoted by $\Phi(a,\sigma)$ is called a *1-transition function* of D if

$$\Phi(a,\sigma) = \{\sigma' : \mathcal{R}es(a,\sigma) \preceq \sigma' \text{ and } \sigma' \setminus \mathcal{R}es(a,\sigma) = K(a,\sigma) \setminus \mathcal{R}es(a,\sigma)\}.$$

□

A 1-model of D is then defined as a pair (σ_0, Φ) where σ_0 is the initial a-state of D and Φ is a 1-transition function of D. The notion of 1-extended function and 1-entailment is then defined as in Definitions 8 and 10 using 1-transition function and 1-model, respectively.

In the next example we shows that the 1-Approximation allows us to reason by cases.

Example 9 Let us consider again the domain D_3 from Example 8. The initial a-state of D_3 is $\sigma_0 = \langle \emptyset, \emptyset \rangle$. It is easy to check that D_3 has one 1-model (σ_0, Φ) with $\Phi(a, \sigma_0) = \{\langle \{f\}, \emptyset \rangle\}$. Hence, we can conclude that $D_3 \models_1 f$ **after** a. □

We now state the relation between the 0-Approximation and the 1-Approximation of domain descriptions of $\mathcal{A}_K$.

Proposition 4 (Soundness of $\models_0$ w.r.t. $\models_1$) Let D be a domain description, φ be a fluent formula of D, and c be a conditional plan. Then,

$$\text{if } D \models_0 \varphi \textbf{ after } c \text{ then } D \models_1 \varphi \textbf{ after } c.$$

Proof. *(Sketch)* It is easy to see that for any domain description D the 0-model and 1-model of D are uniquely determined and the initial a-state in both approximations is also unique. Let us denote them by (σ_0, Φ_0) and (σ_0, Φ_1) respectively. Let σ and δ be two a-states of D such that $\sigma \preceq \delta$. Then, for every action a of D, we can prove that

(i) for each $\sigma' \in \Phi_0(a,\sigma)$ there exists a $\delta' \in \Phi_1(a,\delta)$ such that $\sigma' \preceq \delta'$;

(ii) for each $\delta' \in \Phi_1(a,\delta)$ there exists a $\sigma' \in \Phi_0(a,\sigma)$ such that $\sigma' \preceq \delta'$.

Using (i) and (ii) we can then prove that for any conditional plan c such that $\bot \notin \hat{\Phi}_0(c,\sigma)$,

(iii) for each $\sigma' \in \hat{\Phi}_0(c,\sigma)$ there exists a $\delta' \in \hat{\Phi}_1(c,\delta)$ such that $\sigma' \preceq \delta'$;

(iv) for each $\delta' \in \hat{\Phi}_1(c,\delta)$ there exists a $\sigma' \in \hat{\Phi}_0(c,\sigma)$ such that $\sigma' \preceq \delta'$.

The proposition follows from (iii), (iv), and the fact that $\sigma_0 \preceq \sigma_0$. □

The next example shows that the 1-Approximation is also not able to make some intuitive conclusions[3] that can be made using the $\mathcal{A}_K$ semantics.

Example 10 Consider the domain description:

$$\left.\begin{array}{l} a \textbf{ causes } p \textbf{ if } r \\ a \textbf{ causes } q \textbf{ if } \neg r \\ b \textbf{ causes } f \textbf{ if } p \\ b \textbf{ causes } f \textbf{ if } q \end{array}\right\} = D_4$$

[3] We thank the anonymous AAAI97 reviewer who pointed this out.

The initial a-state is $\langle\emptyset,\emptyset\rangle$, where p,q,r, and f are unknown. Although intuitively and also according to the semantics of $\mathcal{A}_K$, after executing a followed by b in the initial a-state, f should be true, our 1-Approximation is not able to capture this. This is because, the 1-Approximation reasons by cases only up to 1-level. Since after reasoning by cases for 1-level, it summarizes its reasoning to a pair $\langle T,F\rangle$ of sets, it is not able to capture the fact that after executing a in the initial a-state $p\vee q$ is true. □

In the next section, we define an approximation which allows reasoning by cases for multiple levels.

3.3 ω-Approximation

Given a sequence of actions $\alpha = a_1;\ldots;a_n$, we define the prefix of α, denoted by $pre(\alpha)$, as

- if a_1 is a sensing action then $pre(\alpha) = a_1$; or
- if α does not contain a sensing actions then $pre(\alpha)=\alpha$; or
- if a_j is the first sensing action in α, $1 < j \leq n$, then $pre(\alpha) = a_1,\ldots,a_{j-1}$.

The sequence of actions obtained from α after removing its prefix $pre(\alpha)$ is called the *remainder* of α and is denoted by $rem(\alpha)$.

For a sequence of non-sensing actions $\alpha = a_1,\ldots,a_n$, we define

$$\mathcal{R}es(\alpha,\sigma) = \bigcap_{\sigma'\in Comp(\sigma)} Res(a_n, Res(a_{n-1},\ldots,Res(a_1,\sigma')))$$

The ω-Approximation of D is defined by a function Φ_ω, called ω-transition function, which maps a pair of a sequence of actions α and an a-state σ into a set of a-states $\Phi_\omega(\alpha,s)$ as follows.

$$\Phi_\omega(\alpha,\sigma) = \begin{cases} \{\mathcal{R}es(\alpha,\sigma)\} \text{ if } pre(\alpha)=\alpha \text{ and } \alpha \text{ does not contain a sensing action;} \\ \\ \{\sigma' | \sigma' \preceq \mathcal{R}es(a,\sigma), \sigma'\setminus \mathcal{R}es(a,\sigma) = K(a,\sigma)\setminus \mathcal{R}es(a,\sigma)\} \text{ if} \\ \qquad\qquad\qquad\qquad \alpha = a, \text{ and } a \text{ is a sensing action; and} \\ \\ \bigcup_{\sigma'\in\mathcal{R}es(pre(\alpha),\sigma)} \Phi_\omega(rem(\alpha),\sigma'), \text{ otherwise.} \end{cases}$$

An ω-model for a domain D is then defined as the pair (σ_0,Φ_ω), where σ_0 is the initial a-state of D and Φ_ω is an ω-transition function of D.

To extend the function Φ_ω over pairs of conditional plans and a-states we need the following observation.

Observation 3.1 Every conditional plan c can be represented as a sequence of conditional plans $c_1;\ldots;c_n$ where

(a) c_i is either a sequence of actions or a case plan; and

(b) for every $i<n$, if c_i is a sequence of actions then c_{i+1} is a case plan. □

From now on, we will often write a conditional plan c as a sequence $c = c_1; \ldots; c_n$ where c_i's satisfy the conditions (a) and (b) of Observation 3.1.

We now continue with our definition of Φ_ω.

For a conditional plan $c = c_1; \ldots; c_n$ and an a-state σ, we define

1. For $c =$ Case
   $\varphi_1 \rightarrow c_1$
   $\ldots$
   $\varphi_n \rightarrow c_n$
   Endcase,

$$\Phi_\omega(c, \sigma) = \begin{cases} \Phi_\omega(c_i, \sigma) \text{ if } \varphi_i \text{ holds in } \sigma \\ \\ \{\perp\} \quad \text{if none of } \varphi_1, \ldots, \varphi_n \text{ holds in } \sigma \end{cases}$$

2. For $c = c_1; c_2; \ldots; c_n$,

   (a) if c_1 is a sequence of actions,

$$\Phi_\omega(c, \sigma) = \bigcup_{\sigma' \in \Phi_\omega(c_1, \sigma)} \Phi_\omega(c2; \ldots; c_n, \sigma')$$

   (b) if $c_1 =$ Case
   $\varphi_1 \rightarrow p_1$
   $\ldots$
   $\varphi_m \rightarrow p_m$
   Endcase,

$$\Phi_\omega(c, \sigma) = \begin{cases} \bigcup_{\sigma' \in \Phi_\omega(p_i, \sigma)} \Phi_\omega(c2; \ldots; c_n, \sigma') \text{ if } \varphi_i \text{ holds in } \sigma \\ \\ \{\perp\} \quad \text{if none of } \varphi_1, \ldots, \varphi_m \text{ holds in } \sigma \end{cases}$$

3. $\Phi_\omega(c, \perp) = \{\perp\}$ for every conditional plan c.

The notion of ω-entailment is then defined as in Definition 10 using the ω-model.

The next example shows that this generalization indeed overcomes the problem of 1-Approximation in Example 10, through reasoning by cases for multiple levels.

Example 11 Let us consider the domain description D_4 from the Example 10. It is easy to see that $f \in T'$ for every $\langle T', F' \rangle \in \Phi_\omega([a; b], \langle T, F \rangle)$ where $\langle T, F \rangle$ is a complete a-state. Thus, $D_4 \models_\omega f$ **after** $[a; b]$. □

We prove the soundness of $\models_1$ w.r.t. $\models_\omega$ in the next proposition.

Proposition 5 (Soundness of $\models_1$ w.r.t. $\models_\omega$) Let D be a domain description, φ be a formula over fluents of D, and c be a conditional plan. Then,

$$\text{if } D \models_1 \varphi \textbf{ after } c \text{ then } D \models_\omega \varphi \textbf{ after } c.$$

Proof. The proof is similar to the proof of Proposition 4. □

3.4 Soundness of 0, 1 and ω-Approximations w.r.t. $\mathcal{A}_K$-semantics

In the previous sub-sections we discussed three different approximations of $\mathcal{A}_K$. Our next goal is to show that these approximations are sound w.r.t. $\mathcal{A}_K$. Since we have already shown in Propositions 4 and 5 that $\models_0$ is sound w.r.t. $\models_1$ and $\models_1$ is sound w.r.t. $\models_\omega$, we will now show that the ω-Approximation is sound w.r.t. $\mathcal{A}_K$.

Proposition 6 (Soundness of $\models_\omega$ w.r.t. $\models_{\mathcal{A}_K}$) Let D be a domain description, φ be a fluent formula, and c be a conditional plan. Then,

$$\text{if } D \models_\omega \varphi \textbf{ after } c \text{ then } D \models_{\mathcal{A}_K} \varphi \textbf{ after } c.$$

Proof. In Appendix B. $\square$

Even though, ω-Approximation can reason more than the 1-Approximation, it still can not match the $\mathcal{A}_K$ semantics. The following example illustrates this.

Example 12 Let D_5 be the following domain description.

$$\left.\begin{array}{l} a \textbf{ causes } \neg p \textbf{ if } r \\ b \textbf{ determines } r \\ c \textbf{ causes } p \textbf{ if } r \\ \textbf{initially } p \end{array}\right\} = D_5$$

We have that $\sigma_0 = \langle \{p\}, \emptyset \rangle$ is the initial a-state of D_5.

Let $\alpha = [a; b; c]$.

We have that $D_5 \not\models_\omega p \textbf{ after } [a; b; c]$ and $D_5 \not\models_\omega \neg p \textbf{ after } [a; b; c]$.

On the other hand, we can easily check that $D_5 \models^r_{\mathcal{A}_K} p \textbf{ after } [a; b; c]$. $\square$

3.5 Complexity of progression

In this sub-section we will compare the complexity of progression in the various approximations. Suppose the number of fluents we have is n, and d is the size of the domain description. Given an a-state $\langle T, F \rangle$, such that the size of $T \cup F$ is m, the complexity of computing Res and $\mathcal{R}es$ in the different approximations are as follows:

- 0-Approximation: The complexity of computing $Res(a, \sigma)$ is $m\times$ number of ef-propositions in the domain description. This is of the order of $m \times d$.
- 1-Approximation: Here we need to compute $\mathcal{R}es$. This is of the order of $2^{n-m} \times m \times d$.
- ω-Approximation: Here we also need to compute $\mathcal{R}es$. This is also of the order of $2^{n-m} \times m \times d$.

The transition function Φ is defined in terms of Res in case of the 0-Approximation and is defined in terms of $\mathcal{R}es$ in case of the other two approximations. In all of them, if a sensing action a determines p fluents, then when executed in a state σ where non of these fluents are known $\Phi(a, \sigma)$ will have 2^p states.

From the above analysis, it is clear that progression can be done much faster in the 0-Approximation than in the other two. On the other hand there is no significant difference in doing progression between 1-Approximation and ω approximation.

We now show how the 0-Approximation can be computed using logic programs.

3.6 Computing 0-Approximation using logic program

We present a sound and complete translation from a domain description D in $\mathcal{A}_K$ into a disjunctive logic program $\pi_0(D)$ w.r.t the 0-entailment relationship in D.

Given a domain description D in $\mathcal{A}_K$ the disjunctive logic program $\pi_0(D)$ corresponding to D will contain following rules.

1. **Inertia rule:**

$$Holds(F, res(A, S)) \leftarrow Holds(F, S), not\ ab(F, A, S) \tag{3.1}$$

2. **Translating the a-state:**

$$holds(f, s_0). \quad \text{if } \textbf{initially} \quad f \in D,\ f \text{ is a fluent} \tag{3.2}$$

$$\neg holds(f, s_0). \quad \text{if } \textbf{initially} \quad \neg f \in D,\ f \text{ is a fluent} \tag{3.3}$$

   These two rules describe the initial a-state of D.

3. **Translating ef-propositions:** for every ef-proposition a **causes** F **if** $p_1, \ldots, p_n$, the following rules are added to $\pi_0(D)$.

$$Holds(F, res(a, S)) \leftarrow Holds(p_1, S), \ldots, Holds(p_n, S) \tag{3.4}$$

$$ab(\bar{F}, a, S) \leftarrow not\ \overline{Holds(p_1, S)}, \ldots, not\ \overline{Holds(p_n, S)} \tag{3.5}$$

4. **Translating k-propositions:**
   for every k-proposition a **determines** f, the program $\pi_0(D)$ will contain the following rule.

$$holds(f, res(a, S)) \vee \neg holds(f, res(a, S)) \tag{3.6}$$

The following proposition relates $\pi_0(D)$ to D w.r.t. queries.

Proposition 7 (Soundness and Completeness of $\pi_0(D)$ w.r.t. $\models_0$) Let D be a domain description. Then, for every fluent name f and a sequence of actions α in D,

1. $\pi_0(D) \models holds(f, res(\alpha, s_0))$ iff $D \models_0 f$ **after** α;
2. $\pi_0(D) \models \neg holds(f, res(\alpha, s_0))$ iff $D \models_0 \neg f$ **after** α.

Proof. In Appendix E. □

We would like to point out that our attempt to compute 1-Approximation using logic programs was unsuccessful in the sense that we could only come up with a sound (but not complete) translation. (We presented this translation in [BS97].)

4 Related research

4.1 Relationship with Scherl and Levesque's formulation

In Section 2.3 we gave a translation of domain descriptions in D to a first-order theory that used Scherl and Levesque's [SL93] successor-state axiom (which is based on Moore's [Moo85] formulation) and showed the equivalence w.r.t. queries in the language of $\mathcal{A}_K$. Since Scherl and Levesque directly formalize in first-order logic, their formulation is more general than ours; (i) in terms of allowing more general descriptions about the domain such as being able to choose which modal logic to use, and observations about non-initial situations; and (ii) in terms of allowing more general queries.

On the other hand our goal in this paper has been to have a simpler formulation, perhaps at the cost of generality. For example, the 'state' of the agent's knowledge in Scherl and Levesque's formulation (and also in Moore's formulation) would be a Kripke-model. Since planning in a state-space where a 'state' is a Kripke-model is more difficult, we have a simpler notion of a 'state' which we call a k-state. (For instance, if we have n fluents then the number of different Kripke models are $2^{2^{2n}+n}$, while the number of different k-states are 2^{2^n+n}.) As mentioned earlier, our k-state has two components, the real state of the world and the set of possible states that the agent thinks it may be in. Our k-state is actually equivalent to a Kripke-model when we consider the logic S5. Thus with a goal to make things simpler we sacrifice generality and make an a-priori decision on which logic of knowledge to use.

Also, since we develop a high level language $\mathcal{A}_K$, with an independent semantics – that does not depend on standard logics, it can serve the role of a benchmark for languages with sensing actions, at least for the restricted class of queries in $\mathcal{A}_K$. Moreover, this high level language makes it easier for us to prove the soundness of approximations that have a much less and more manageable state-space. By having sound and complete translations of domain descriptions in $\mathcal{A}_K$ to theories in first-order logic that use Scherl and Levesque's axioms, our sound approximations are also in away sound approximations of Scherl and Levesque's formalism.

Finally, we would like to mention that loop-free robot programs of [Lev96] are special cases of our conditional plans. In particular, the statements $seq(a, r)$ and $branch(a, r_1, r_2)$ of [Lev96] can be recursively translated to conditional plans a, r and

a
Case
  $f \rightarrow r_1$
  $\neg f \rightarrow r_2$
Endcase

respectively. In this paper we do not allow loops in our conditional plans. But the ideas in [Lev96, LTM97] can be used to extend our conditional plans to allow loops.

4.2 Relationship with Lobo et al.'s semantics

Lobo et al. in [LTM97] have a goal similar to ours, in terms of developing a high-level language that allows sensing actions and giving translations of it to theories in a standard logical language. We now list some of the major differences between both approaches:

- They represent the state of an agent's knowledge by a set of states (which they refer to as a situation), and the transition function Φ in their model is defined such that for a sensing action a and a situation Σ, $\Phi(a, \Sigma)$ is a subset of Σ that consists of all states which agree on the fluent values determined by the sensing action a. A drawback of this approach is that domain descriptions have a lot of models. But more importantly, it is possible that when a domain description has two actions a and b that determine the same fluent f, there are models Φ, such that $\Phi(a, \Sigma) \neq \Phi(b, \Sigma)$ for some Σ's. In other words, while f may be true in all states in $\Phi(a, \Sigma)$, it may be false in all states in $\Phi(b, \Sigma)$. We find such models unintuitive.

- The semantics of $\mathcal{A}_K$ is more general than the semantics of Lobo et al. in the sense that in their formulation the assumption about models being rational is hard wired into the semantics.
- On the other hand the high level language used by Lobo et al. is more general than the one we are using. They allow preconditions in k-propositions. We do not allow preconditions in k-propositions, as it is not clear to us whether these preconditions should be evaluated with respect to the world or w.r.t. the agent's knowledge about the world. For example, suppose we have a k-proposition of the form a **determines** f **if** p. Now suppose in a situation the agent has no knowledge about p and it executes a. In Lobo et al.'s characterization, in the resulting state the agent could be in three different state of mind. It will either know that p is true and f is true; or it will know that p is true and f is false; or it would know that p is false. Thus in their formulation, after the execution of a the agent would know the truth value of p. We feel that if this is the case, then the agent should have a direct k-proposition saying a **determines** p. Nevertheless, we can capture the meaning attributed by Lobo et al. to the k-proposition a **determines** f **if** p, by expanding our language in a straight forward manner (See [BS97] for details) to allow preconditions in k-propositions and then by breaking a into two actions a_1 and a_2 and having a_1 **determines** p and a_2 **determines** f **if** p.
- Lobo et al. give translations of their domain descriptions to theories in epistemic logic programs [Gel91]. We give translations to disjunctive logic programs, which are simpler than epistemic logic programs. We also give translations to first-order theories.
- Finally, we consider sound approximations of our language. In the later part of this section we show our semantics to be equivalent (sometimes) to theirs. Thus our approximations are also sound approximations of their formulation.

We now give a quick overview of the formulation in [LTM97], where we only consider domain descriptions in the syntax of $\mathcal{A}_K$ in this paper. We then show that our rational semantics is equivalent to the semantics in [LTM97] for this restricted case. The semantics of [LTM97] is defined through transition functions that map pairs of actions and situations into situations where a situation is a set of states. A situation is consistent if it is not empty. Given a domain description D, the situation consisting of all the initial states of D, denoted by Σ_0, is called the *initial situation* of D. A fluent f is said to be *true* in a situation Σ if $f \in s$ for every $s \in \Sigma$. A formula fluent φ is said to be *true* in a situation Σ if φ is true in every state s belonging to Σ. We will need the following definition.

Definition 12 Let Σ be a consistent situation and f a fluent. A consistent situation Σ' is "*f-compatible*" with Σ iff

1. $\Sigma' = \{\sigma \in \Sigma : f \notin \sigma\}$; or
2. $\Sigma' = \{\sigma \in \Sigma : f \in \sigma\}$

For a domain description D, a function Φ from pairs of actions and situations into situations is called an interpretation of D.

Definition 13 An interpretation Φ of a domain description D is a *model* of D if and only if

1. for any consistent situation Σ.

   (a) for any non-sensing action a,

   $$\Phi(a, \Sigma) = \bigcup_{\sigma \in \Sigma} \{Res(a, \sigma)\}.$$

   (b) for each sensing action a_s, let

   $$\begin{array}{l} a_s \textbf{ determines } f_1 \\ \quad\ldots \\ a_s \textbf{ determines } f_n \end{array}$$

   be the k-propositions in which a_s occurs. Then,

   - $\Phi(a_s, \Sigma)$ must be a consistent situation; and
   - $\Phi(a_s, \Sigma) = \Sigma_1 \cap \Sigma_2 \ldots \cap \Sigma_n$ where Σ_i is a f_i-compatible situation with Σ for $i = 1, \ldots, n$

2. for any action a, $\Phi(a, \emptyset) = \emptyset$.

Lobo et al. extend the function Φ to a plan evaluation function $\Gamma_\Phi(c, \Sigma)$ which allows conditional plans. The definition of $\Gamma_\Phi(c, \Sigma)$ given in [LTM97] is very similar to the definition of $\hat{\Phi}$ and we omit it here for brevity. In the following example, we show the difference between our models and the models of Lobo et al.

Example 13 Let us consider the domain description D_1 from Example 1. The states of D_1 are:

$$\begin{array}{ll} s_1 = \emptyset. & s_5 = \{disarmed\} \\ s_2 = \{locked\} & s_6 = \{disarmed, exploded\} \\ s_3 = \{exploded\} & s_7 = \{disarmed, locked\} \\ s_4 = \{locked, exploded\} & s_8 = \{disarmed, locked, exploded\} \end{array}$$

The initial situation of D_1 is $\Sigma_0 = \{s_1, s_2\}$. There are two *locked*-compatible situations with Σ_0: $\Sigma_1 = \{s_1\}$ and $\Sigma_2 = \{s_2\}$. Thus, if Φ is a model of D_1, then either $\Phi(look, \Sigma_0) = \{s_1\}$ or $\Phi(look, \Sigma_0) = \{s_2\}$, i.e., in the approach of Lobo et al. there are (at least) two different models which differ from each other by the transition functions. On the other hand, in our approach we have two rational models which differ only by the initial k-states. □

The entailment relation w.r.t. Lobo et al.'s semantics is defined next.

Definition 14 $D \models_{LTM} \varphi$ **after** c iff for every model Φ of D, φ is true in $\Gamma_\Phi(c, \Sigma)$.

The following proposition relates Lobo et al's semantics with ours.

Proposition 8 (Equivalence between $\models^r_{\mathcal{A}_K}$ and $\models_{LTM}$) Let D be a domain description, φ be a fluent formula in D, and c be a conditional plan in D. Then,

$$D \models^r_{\mathcal{A}_K} \varphi \textbf{ after } c \quad \text{iff} \quad D \models_{LTM} \varphi \textbf{ after } c.$$

Proof. In Appendix D. □

5 Conclusion

In this paper we present a high-level action description language that takes into account sensing actions and distinguishes between the state of the world and the state of the knowledge of an agent about the world. We give sound and complete translation of domain descriptions in our language to theories in first-order logic and disjunctive logic programming. We compare our formulation with others and analyze the state-space of our formulation and that of the others. We then give sound approximations of our formulation with a much smaller state-space. We believe the approximations in this paper will be very important in developing practical planners. Our future goal is to analyze existing planners[4] that construct conditional plans and use sensing actions and develop more efficient planners based on the approximations described in this paper. We have made a head start in this direction by implementing a simple generate and test planner in Prolog.

References

[AB91] K. Apt and M. Bezem. Acyclic programs. *New Generation Computing*, 9(3,4):335–365, 1991.

[Bar95] C. Baral. Reasoning about Actions : Non-deterministic effects, Constraints and Qualification. In *Proc. of IJCAI 95*, pages 2017–2023, 1995.

[BG94] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994.

[BG97] C. Baral and M. Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31(1-3):85–117, May 1997.

[BGP97] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, May 1997.

[BS97] C. Baral and T. Son. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. of International Logic Programming Symposium (ILPS 97)*, pages 387–401, 1997.

[BS98] C. Baral and T. Son. Formalizing sensing actions: a transition function based approach. Technical report, Dept of Computer Science, University of Texas at El Paso (http://cs.utep.edu/chitta/chitta.html), 1998.

[CSW95] W. Chen, T. Swift, and D. Warren. Efficient top-down computation of queries under the well-founded semantics. *Journal of Logic Programming*, 24(3):161–201, 1995.

[DMB92] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proc. of ECAI92*, 1992.

[Dun93] P. Dung. Representing actions in logic programming and its application in database updates. In D. S. Warren, editor, *Proc. of ICLP-93*, pages 222–238, 1993.

[4] From the following quote in [GW96]:
"In UWL (and in SADL) individual literals have truth values expressed in three valued logic: T, F, U (unknown). " it seems that they are using an approximation. We would like to analyze this planner to figure out what kind of approximation they are using and if it is sound w.r.t. one of the formulations discussed in this paper.

[EHW+92] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *KR 92*, pages 115–125, 1992.

[Gel91] M. Gelfond. Strong introspection. In *Proc. AAAI-91*, pages 386–391, 1991.

[GEW96] K. Golden, O. Etzioni, and D. Weld. Planning with execution and incomplete informations. Technical report, Dept of Computer Science, University of Washington, TR96-01-09, February 1996.

[GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, pages 365–387, 1991.

[GL92] M. Gelfond and V. Lifschitz. Representing actions in extended logic programs. In *Joint International Conference and Symposium on Logic Programming.*, pages 559–573, 1992.

[GL93] M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323, 1993.

[GW96] K. Golden and D. Weld. Representing sensing actions: the middle ground revisited. In *KR 96*, pages 174–185, 1996.

[Haa86] L. Haas. A syntactic theory of belief and action. *Artificial Intelligence*, 28:245–292, May 1986.

[Kar93] G. Kartha. Soundness and completeness theorems for three formalizations of action. In *IJCAI 93*, pages 724–729, 1993.

[KL94] G. Kartha and V. Lifschitz. Actions with indirect effects: Preliminary report. In *KR 94*, pages 341–350, 1994.

[KOG92] K. Krebsbach, D. Olawsky, and M. Gini. An empirical study of sensing and defaulting in planning. In *First Conference of AI Planning Systems*, pages 136–144, 1992.

[Lev96] H. Levesque. What is planning in the presence of sensing? In *AAAI 96*, pages 1139–1146, 1996.

[Lif95] V. Lifschitz. Nested Abnormality Theories. *Artificial Intelligence*, 74:351–365, 1995.

[Lif97] V. Lifschitz. Two components of an action language. *Annals of Math and AI*, 21(2-4):305–320, 1997.

[LT94] V. Lifschitz and H. Turner. Splitting a Logic Program. *ICLP'94*, MIT Press, 1994.

[LTM97] J. Lobo, S. Taylor, and G. Mendez. Adding knowledge to the action description language $\mathcal{A}$. In *AAAI 97*, pages 454–459, 1997.

[MS89] W. Marek and V.S. Subrahmanian. The relationship between Stable, Supported, Default and Auto-Epistemic Semantics for General Logic Programs. *ICLP'89*, pages 600–620, 1989.

[Moo79] R. Moore. *Reasoning about knowledge and action.* PhD thesis, MIT, Feb 1979.

[Moo85] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal theories of the commonsense world.* Ablex, Norwood, NJ, 1985.

[Rei91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[Sha97] M. Shanahan. *Solving the frame problem: A mathematical investigation of the commonsense law of inertia.* MIT press, 1997.

[SL93] R. Scherl and H. Levesque. The frame problem and knowledge producing actions. In *AAAI 93*, pages 689–695, 1993.

[Tur94] H. Turner. Signed logic programs. In *Proc. of the 1994 International Symposium on Logic Programming*, pages 61–75, 1994.

[Tur97] H. Turner. Representing actions in logic programs and default theories. *Journal of Logic Programming*, 31(1-3):245–298, May 1997.

Appendix A

In this section, we prove the propositions 1 and 2. We will need the following notations and lemmas.

Let c be a conditional plan, we define the number of case plans of c, denoted by $count(c)$, inductively as follows.

1. If $c = []$, then $count(c) = 0$.
2. If $c = a$, a is an action, then $count(c) = 0$
3. If c_1 and c_2 are conditional plans, then $case(c_1; c_2) = count(c_1) + count(c_2)$
4. If c is a case plan of the form

   Case
   $\varphi_1 \rightarrow c_1$
   ...
   $\varphi_n \rightarrow c_n$
   Endcase

   then $count(c) = 1 + \Sigma_{i=1}^{n} count(c_i)$.

Let $A_1, \ldots, A_l$ be actions of D and S be a situation term of $R(D)$. By $do([A_1, \ldots, A_l], S)$ we denote the term

$$do(A_l, do(A_{l-1}, \ldots, do(A_1, S))).$$

Throughout the section, if not otherwise specified, we assume that D is a domain description with $\mathbf{m}$ value propositions **initially** G_1, ..., **initially** $G_{\mathbf{m}}$ and $\mathbf{n}$ sensing actions $K_1, \ldots, K_{\mathbf{n}}$ with the k-propositions K_1 **determines** F_1, ..., $K_{\mathbf{n}}$ **determines** $F_{\mathbf{n}}$.

We first prove the following lemmas.

Lemma 2.1 *Let D be a domain description and $M = (\sigma_0, \Phi)$ be a model of D. Then, there exists a model M_R of $R(D)$ such that for any fluent F and any sequence of actions α of D,*

1. *F is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$ iff $M_R \models Knows(F, do(\alpha, S_0))$;*

2. *F is known to be false in $\hat{\Phi}(\alpha, \sigma_0)$ iff $M_R \models Knows(\neg F, do(\alpha, S_0))$.*

Proof. We will construct a model M_R of $R(D)$ satisfying the lemma. First, we define the different universes: the universe of fluents $|M_R|_{fluent}$, the universe of actions $|M_R|_{action}$, and the universe of situations $|M_R|_{situation}$.

1. $|M_R|_{action} = \mathbf{A}$ where $\mathbf{A}$ is the set of actions of D;

2. $|M_R|_{fluent} = \mathbf{F}$ where $\mathbf{F}$ is the set of fluents of D; and

3. $|M_R|_{situation} = \{\langle s, \Sigma\rangle : s \subseteq \mathbf{F}, \Sigma \subseteq 2^{\mathbf{F}}\}$ which is the set of k-states of D.

Fluent constants and action constants are interpreted as themselves. To complete the definition of M_R we need to define the interpretation of the situation constant S_0, the predicates $Holds$ and K, and the function do. Let us denote the interpretation of $Holds$, K, and do by $Holds^{M_R}$, K^{M_R}, and do^{M_R}, respectively. For objects o of type situation, let us denote o's interpretation by σ_o. In the following, we write $\sigma.1$ and $\sigma.2$ to denote the first and second component of a k-state σ, respectively. I.e., if $\sigma = \langle s, \Sigma\rangle$, then $\sigma.1$ denotes s and $\sigma.2$ denotes Σ. We define

(a) $\sigma_{S_0} = \sigma_0$;

(b) $\langle F, s\rangle \in Holds^{M_R}$ iff $F \in \sigma_s.1$;

(c) $\langle s', s\rangle \in K^{M_R}$ iff $\sigma_{s'}.1 \in \sigma_s.2$; and

(d) $do^M(A, s) = \Phi(A, \sigma_s)$.

(a)-(d) complete the definition of M_R. We will now prove that M_R is a model of $R(D)$. To do that we will prove that M_R satisfies the axioms (2.6)-(2.11).

We have that

$\gamma_F^+(A, s)$ holds in M_R
iff $\bigvee_{X \textbf{ causes } F \textbf{ if } \varrho_X \in A_F^+}(X = A \wedge Holds^{M_R}(\varrho_X, s))$ is true (Definition of $\gamma_F^+(A, s)$)
iff there exists A **causes** F **if** ϱ in D such that $\varrho \subseteq \sigma_s.1$ (by (b))
iff $F \in E_A^+(\sigma_s.1)$ (1)

Similarly, we can prove that $\gamma_F^-(A, s)$ holds in M_R iff $F \in E_A^-(\sigma_s.1)$ (2)

We have that
$Holds(F, do(A, s))$ holds in M_R
iff $F \in \Phi(A, \sigma_s).1$ (by (b) and (d))
iff $F \in Res(A, \sigma_s.1)$ (by definition of Φ)
iff $F \in E_A^+(\sigma_s.1)$
or $F \in \sigma_s.1 \wedge F \notin E_A^-(\sigma_s.1)$ (by definition of Res)
iff $\gamma_F^+(A, s)$ holds in M_R
or $Holds(F, s) \wedge \gamma_F^-(A, s)$ holds in M_R (by (1), (2) and (b))
iff $\gamma_F^+(A, s) \vee (Holds(F, s) \wedge \gamma_F^-(A, s))$ holds in M_R
Thus M_R satisfies (2.6). (3)

Similarly, we can prove that M_R satisfies (2.7). (4)

We now prove that M_R also satisfies (2.8)-(2.11).

Since M is a model of D, we have that if s is a situation satisfying $\sigma_s.1 \in \sigma_{S_0}.2$, then for $i = 1, \ldots, \mathbf{m}$, G_i holds in $\sigma_s.1$. This implies that for every s, $\sigma_s.1 \in \sigma_{S_0}.2$, $\bigwedge_{i=1}^{\mathbf{m}} Holds(G_i, s)$ holds in M_R. (*)

Since σ_0 is a grounded initial k-state of D, we have that $\sigma_{S_0}.1 \in \sigma_{S_0}.2$. Hence, from (*), we can conclude that $\bigwedge_{i=1}^{\mathbf{m}} Holds(G_i, S_0)$ holds in M_R. This proves that (2.9) is satisfied by M_R. (5)

Furthermore, from $\sigma_{S_0}.1 \in \sigma_{S_0}.2$ and (c), we conclude that $K(S_0, S_0)$ holds in M_R. Hence, M_R satisfies (2.11). (6)

Assume that $K(s, S_0)$ holds in M_R. From (c), we have that $\sigma_s.1 \in \sigma_{S_0}.2$. Hence, by (*), we conclude that $\bigwedge_{i=1}^{\mathbf{m}} Holds(G_i, s)$ holds in M_R. This proves that (2.10) is satisfied by M_R. (7)

It remains to be shown that M_R satisfies (2.8). We consider two cases:

- **Case 1:** $K(s'', do(A, s))$ holds in M_R. We will prove that

  (e) $\exists s'[(K(s', s) \wedge s'' = do(A, s')) \wedge$
  $((\wedge_{i=1}^{\mathbf{n}} A \neq K_i) \vee \bigvee_{i=1}^{\mathbf{n}} (A = K_i \wedge Holds(F_i, s') \equiv Holds(F_i, s))]$ holds in M_R.

  From the assumption that $K(s'', do(A, s))$ holds in M_R and Items (c) and (d) of the Definition of M_R we have that

  $\sigma_{s''}.1 \in \Phi(A, \sigma_s).2.$ (a1)

  This implies that there exists a σ such that

  $\sigma.1 \in \sigma_s.2$, and (a2)

  $\sigma_{s''} = \Phi(A, \sigma).$ (a3)

  As $\sigma \in |M_R|_{situation}$ we can conclude that there exists a situation s' such that $\sigma_{s'} = \sigma$. We will prove that s' satisfies (e).

  From (a2) and (c), and because of $\sigma_{s'} = \sigma$, we have that $K(s', s)$ holds in M_R. (a4)

  From (a3), (d), and $\sigma_{s'} = \sigma$, we have that $s'' = do(A, s')$ holds in M_R. (a5)

  Here, we have two sub-cases:

  Case A: A is a non-sensing action. Then, we have that $\bigwedge_{i=1}^{\mathbf{n}} A \neq K_i$ holds in M_R. (a6)

  Case B: A is a sensing action, say $A = K_j$. Then, from (a1), and the definition of Φ, we have that $F_j \in \sigma_{s''}.1$ iff $F_j \in \Phi(A, \sigma_s).1$. (a7)

  From (a3) and $\sigma_{s'} = \sigma$, we have that $F_j \in \sigma_{s''}.1$ iff $F_j \in \Phi(A, \sigma_{s'}).1$. (a8)

  From (a7) and (a8) we have that $F_j \in \Phi(A, \sigma_{s'}).1$ iff $F_j \in \Phi(A, \sigma_s).1$.
  Hence, by definition of Φ, $Holds(F_j, s') \equiv Holds(F_j, s)$. This implies that
  $\bigvee_{i=1}^{\mathbf{n}} (A = K_i \wedge Holds(F_j, s') \equiv Holds(F_j, s))$ holds in M_R. (a9)

  From (a6) and (a9) we conclude that
  $(\wedge_{i=1}^{\mathbf{n}} A \neq K_i) \vee \bigvee_{i=1}^{\mathbf{n}} (A = K_i \wedge Holds(F_i, s') \equiv Holds(F_i, s))$ holds in M_R.
  This, together with (a4) and (a5), proves (e). (a10)

- **Case 2:**
$\exists s'[(K(s',s) \wedge s'' = do(A,s'))\wedge$
$((\wedge_{i=1}^{\mathbf{n}} A \neq K_i) \vee \bigvee_{i=1}^{\mathbf{n}}(A = K_i \wedge Holds(F_i,s') \equiv Holds(F_i,s))]$ holds in M_R.

  We want to show that

  (f) $K(s'', do(A,s))$ holds in M_R.

  Since $K(s',s)$ holds in M_R we have that $\sigma_{s'}.1 \in \sigma_s.2$. (b1)

  If A is a non-sensing action, then we have that $\Phi(A,\sigma_{s'}).1 \in \Phi(A,\sigma_s).2$. (b2)

  If A is a sensing action, say $A = K_j$, then we have that $Holds(F_i,s') \equiv Holds(F_i,s)$ holds in M_R. This implies that $\Phi(A,\sigma_{s'}).1 \in \Phi(A,\sigma_s).2$. (b3)

  From (b2) and (b3) we can conclude that $\Phi(A,\sigma_{s'}).1 \in \Phi(A,\sigma_s).2$. (b4)

  Since $s'' = do(A,s')$ we have that $\sigma_{s''}.1 = \Phi(A,\sigma_{s'}).1$. Together with (b4), we have that $\sigma_{s''}.1 \in \Phi(A,\sigma_s).2$. Hence, by (c), we have that $K(s'', do(A,s))$. This proves (f). (b5)

It follows from (a10) and (b5) that M_R satisfies (2.8). (8)

We have proved that M_R satisfy the axioms (2.6)-(2.11) of $R(D)$ (by (3)-(8)). Thus, M_R is a model of $R(D)$. We now show that M_R satisfies the conclusion of the lemma.

Using induction over the length of α, we can prove that

(**) $$K(s'', do(\alpha,S_0)) \quad \text{iff} \quad \sigma_{s''}.1 \in \hat{\Phi}(\alpha,\sigma_0).2$$

We have that
F is known to be true in $\hat{\Phi}(\alpha,\sigma_0)$
iff for every σ'' such that $\sigma''.1 \in \hat{\Phi}(\alpha,\sigma_0).2$, F is true in $\sigma''.1$ (by Definition)
iff for every s'' such that $\sigma_{s''} = \sigma''$,
$K(s'', do(\alpha,S_0))$ holds in M_R and $Holds(F,s'')$ holds in M_R (by (**))
iff $Knows(F, do(\alpha,S_0))$ holds in M_R
iff $M_R \models Knows(F, do(\alpha,S_0))$. (9)

Similarly, we can prove that F is known to be false in $\hat{\Phi}(\alpha,\sigma_0)$ iff $M_R \models Knows(\neg F, do(\alpha,S_0))$. (10)

The lemma follows from (9) and (10). $\Box$

Lemma 2.2 *Let D be a domain description and M_R be a model of $R(D)$. Then, there exists a model $M = (\sigma_0, \Phi)$ of D such that for any fluent F and any sequence of actions α of D,*

1. *$M_R \models Knows(F, do(\alpha,S_0))$ iff F is known to be true in $\hat{\Phi}(\alpha,\sigma_0)$;*
2. *$M_R \models Knows(\neg F, do(\alpha,S_0))$ iff F is known to be false in $\hat{\Phi}(\alpha,\sigma_0)$.*

Proof. Let M_R be a model of $R(D)$. Since $R(D)$ contains the DCA and UNA axioms for actions and fluents we can assume that the domains of actions and fluents are $\mathbf{A}$ and $\mathbf{F}$ respectively, i.e., $|M_R|_{action} = \mathbf{A}$ and $|M_R|_{fluent} = \mathbf{F}$. For each situation term s in M_R, let $s^{M_R} = \{F : Holds(F,s) \in M_R\}$. We construct $M = (\sigma_0, \Phi)$ as follows.

(a1) $\sigma_0.1 = S_0^{M_R}$;

(a2) $\sigma_0.2 = \{s^{M_R} : K(s, S_0) \in M\}$.

(a3) Φ is defined as in the Definition 2.

We have that
$\gamma_F^+(A, s)$ holds in M_R
iff $\bigvee_{X \textbf{ causes } F \textbf{ if } \varrho_X \in D}(X = A \wedge Holds(\varrho_X, s))$ holds in M_R (by Definition of $\gamma_F^+(A, s)$)
iff there exists A **causes** F **if** ϱ in D such that $Holds(\varrho, s)$ holds in M_R
iff there exists A **causes** F **if** ϱ in D such that $\varrho \subseteq s^{M_R}$
iff $F \in E_A^+(s^{M_R})$ (1)

Similarly, from Definition of $\gamma_F^-(A, s)$ we can prove that $\gamma_F^-(A, s)$ holds in M_R iff $F \in E_A^-(s^{M_R})$. (2)

From (2.9) we have that for $i = 1, \ldots, \mathbf{m}$, $Holds(G_i, S_0)$ holds in M_R. Hence, by (a), we have that $G_i \in \sigma_0.1$, for $i = 1, \ldots, \mathbf{m}$. In other words, we have that

$\sigma_0.1$ is an initial state of D. (3)

Consider $s' \in \sigma_0.2$. By (a2) we have that
$\Rightarrow$ there exists s such that $s' = s^{M_R}$ and $K(s, S_0)$
$\Rightarrow \bigwedge_{i=1}^{\mathbf{m}} Holds(G_i, s)$ holds in M_R (By 2.10)
$\Rightarrow$ for $i = 1, \ldots, \mathbf{m}$, $Holds(G_i, s)$ holds in M_R
$\Rightarrow$ for $i = 1, \ldots, \mathbf{m}$, $G_i \in s^{M_R} = s'$ (by Definition of s^{M_R})
$\Rightarrow s'$ is a initial state of D (4)

From (3) and (4) we have that σ_0 is an initial k-state. From (a3), we know that Φ is a transition function of D. Furthermore, from axiom (2.11) and (a2), we can conclude that $\sigma_0.1 \in \sigma_0.2$, i.e., σ_0 is a grounded initial k-state. Hence, $M = (\sigma_0, \Phi)$ is a model of D.

We now prove by induction over l the following assertion:

(*) *for any sequence of actions* $A_1, \ldots, A_l$,
$K(s, do([A_1, \ldots, A_l], S_0)) \in M$ *iff* $s^{M_R} \in \hat{\Phi}([A_1, \ldots, A_l], \sigma_0).2$.

Base case: $|\alpha| = 0$. Then, we have that $do([], S_0) = S_0$ and $\hat{\Phi}([], \sigma_0) = \sigma_0$. Hence, (*) is equivalent to

$$\forall s\ (K(s, S_0) \in M \quad \text{iff} \quad s^{M_R} \in \sigma_0.2)$$

which is trivial by Item (a2) of the definition of M. This proves the base case.

Inductive step: Assume that (*) is proved for $l - 1$. We need to prove it for l. By axiom 2.8 of $R(D)$ we have that

$K(s, do([A_1, \ldots, A_l], S_0)) \in M_R$
iff $\exists u(K(u, do([A_1, \ldots, A_{l-1}], S_0)) \wedge s = do(A_l, u)$
and $(\bigwedge_{j=1}^{\mathbf{n}}(A_l \neq K_j) \wedge$
$\bigvee_{j=1}^{\mathbf{n}}(A_l = K_j \wedge (Holds(F_j, s) \equiv Holds(F_j, do([A_1, \ldots, A_{l-1}])))))$ holds in M_R (by (2.8))
iff $u^{M_R} \in \hat{\Phi}([A_1, \ldots, A_{l-1}], \sigma_0).2$ (by inductive hypothesis)
and $s^{M_R} = Res(A_l, u^{M_R})$ (by $s = do(A_l, u)$ and (a1))

and if $A_l = K_j$ then $F_j \in u^{M_R}$ iff $F_j \in \hat{\Phi}([A_1, \ldots, A_{l-1}], \sigma_0).1$
iff $s^{M_R} \in \hat{\Phi}([A_1, \ldots, A_l], \sigma_0).2$ (5)

(5) proves that (*) holds for l. Hence, we conclude the inductive step.

We now prove that (σ_0, Φ) satisfies the conclusion of the lemma. For a fluent F we have that

$M \models Knows(F, do(\alpha, S_0))$
iff $\forall s\ K(s, do(\alpha, S_0)) \supset Holds(F, s)$ holds in M_R
iff $\forall s^M \in \hat{\Phi}(\alpha, \sigma_0).2, F \in s^M$ (by (*))
iff F is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$. (6)

Similarly, we can prove that $M \models Knows(\neg F, do(\alpha, S_0))$ iff F is known to be false in $\hat{\Phi}(\alpha, \sigma_0)$. (7)

The lemma is proved by (6) and (7). □

We now prove Proposition 1.

Proposition 1 *Let D be a domain description, φ be a fluent formula, and α be a sequence of actions of D. Then,*

$$D \models_{\mathcal{A}_K} \varphi \textbf{ after } \alpha \quad \textit{iff} \quad R(D) \models Knows(\varphi, do(\alpha, S_0)).$$

Proof. It suffices to prove the proposition for a fluent F.

1. Assume that $D \models_{\mathcal{A}_K} F$ **after** α. We will prove that $R(D) \models Knows(F, do(\alpha, S_0))$. Assume the contrary, $R(D) \not\models Knows(F, do(\alpha, S_0))$. By definition, there exists a model M_R of $R(D)$ such that $M_R \not\models Knows(F, do(\alpha, S_0))$. Then, by Lemma 2.2, there exists a model M of D such that $M \not\models F$ **after** α. This implies that $D \not\models F$ **after** α. This contradicts with the assumption that $D \models F$ **after** α. Hence, our assumption is incorrect, i.e., we have proved that $R(D) \models Knows(F, do(\alpha, S_0))$. Therefore, we can conclude that

   if $D \models_{\mathcal{A}_K} F$ **after** α then $R(D) \models Knows(F, do(\alpha, S_0))$. (1)

2. Assume that $R(D) \models Knows(F, do(\alpha, S_0))$. We will prove that $D \models_{\mathcal{A}_K} F$ **after** α. Assume the contrary, $D \not\models_{\mathcal{A}_K} F$ **after** α. This means that there exists a model M of D such that $M \not\models F$ **after** α. Then, by Lemma 2.1, there exists a model M_R of $R(D)$ such that $M_R \not\models Knows(F, do(\alpha, S_0))$. This implies that $R(D) \not\models Knows(F, do(\alpha, S_0))$. This contradicts our assumption that $R(D) \models Knows(F, do(\alpha, S_0))$. Hence, we have that $D_{\mathcal{A}_K} \models F$ **after** α. We can conclude that

   if $R(D) \models Knows(F, do(\alpha, S_0))$ then $D \models_{\mathcal{A}_K} F$ **after** α. (2)

From (1) and (2), we can conclude that
$D \models_{\mathcal{A}_K} F$ **after** α iff $R(D) \models Knows(F, do(\alpha, S_0))$. □

We will now extend the Lemmas 2.1 and 2.2 to conditional plans. (Recall that from now on $Know(\varphi, S)$ is a shorthand for $S \neq \perp \wedge \forall s'(K(s', S) \supset Holds(\varphi, s'))$.) In the process, we will use the following lemmas:

Lemma 2.3 *The following formulas are entailed by B_{Apply}:*

(i) $Apply([], s, s') \supset s = s'$;

(ii) $Apply([a|\alpha], s, s') \wedge s \neq \perp \supset Apply(\alpha, do(a, s), s')$*;*

(iii) $Apply([case([])|c], s, s') \supset s' = \perp$*;*

(iv) $Apply(c, \perp, s') \supset s' = \perp$*;*

(v) $Apply([case([(\varphi, c)|c'])|c''], s, s') \wedge s \neq \perp \supset$
$$((knows(\varphi, s) \supset \exists s''.Apply(c, s, s'') \wedge Apply(c'', s'', s')) \wedge (\neg knows(\varphi, s) \supset Apply([case(c')|c''], s, s')));$$

(vi) *for every sequence of actions* α*,* $Apply(\alpha, s, s') \wedge s \neq \perp \supset do(\alpha, s) = s'$*;*

(vii) $Apply(c, s, s') \wedge Apply(c, s, s'') \supset s' = s''$*.*

Proof: The proofs of the items (i)-(v) are similar. Item (vi) follows from items (ii) and (i). We present here only the proof of (ii). Assume the contrary, i.e., there exists a model M of B_{Apply} such that (ii) does not hold. This implies that there exists an action a, a sequence of actions α, and two situations s and s' such that

$Apply([a|\alpha], s, s') \wedge s \neq \perp$ holds in M and

$Apply([a|\alpha], do(a, s), s')$ does not hold in M.

By definition of B_{Apply}, the model $M' = M \setminus \{Apply([a|\alpha], s, s')\}$ is a model of B_{Apply}. This contradicts the assumption that M is a minimal model of B_{Apply}. Hence, our assumption that (ii) does not hold in M is incorrect, i.e., we have proved that (ii) is a valid sentence of B_{Apply}.

The proof of Item (vii) is based on induction over $count(c)$ and is omitted here. □

Lemma 2.4 *Let* $c_1, \ldots, c_n$ *be* n *arbitrary conditional plans,* $(n \geq 1)$*. Then, the following formula is entailed* B_{Apply}*:*

$$Apply([c_1; \ldots; c_n], s_1, s_{n+1}) \equiv \exists(s_2, \ldots, s_n).Apply(c_1, s_1, s_2) \wedge \ldots \wedge Apply(c_n, s_n, s_{n+1}).$$

Proof. We prove the lemma by induction over n.

Base case: $n = 1$. Then, we have that the right hand side is $\exists s_2.Apply(c_1, s_1, s_2)$ and the left hand side is $Apply(c_1, s_1, s_2)$. It follows from Item (vii) of Lemma 2.3 that $Apply(c_1, s_1, s_2) \equiv \exists(s_2).Apply(c_1, s_1, s_2)$. This proves the base case.

Inductive step: Assume that we have proved the lemma for n. We need to prove it for $n + 1$. Since c_n and c_{n+1} are conditional plans, by definition, we have that $c = c_n; c_{n+1}$ is a conditional plan. Hence, by inductive hypothesis for n plans $c_1, \ldots, c_{n-1}, c$, we have that

$$Apply([c_1; \ldots; c_{n-1}; c], s_1, s_{n+2}) \equiv \exists(s_2, \ldots, s_n).Apply(c_1, s_1, s_2) \wedge \ldots \wedge Apply(c, s_n, s_{n+2}). \quad (1)$$

By Inductive hypothesis for 2 plans c_n and c_{n+1}, we have that

$$Apply(c, s_n, s_{n+2}) \equiv \exists(s_{n+1}).Apply(c_n, s_n, s_{n+1}) \wedge Apply(c_{n+1}, s_{n+1}, s_{n+2}) \quad (2)$$

The inductive step follows from (1) and (2). I.e., the lemma is proved. □

Lemma 2.5 *Let* c *be a case plan of the form*

Case
  $\varphi_1 \rightarrow p_1$

$\vdots$

$\varphi_l \to p_l$

Endcase

and $s \neq \perp$. *Then, the following formula is entailed by* B_{Apply}.

$Knows(\varphi_j, s) \wedge Apply(c, s, s') \equiv Knows(\varphi_j, s) \wedge Apply(p_j, s, s')$.

Proof. Let M be a model of B_{Apply}. We consider two cases:

(i) **Left to Right:** Assume that $Knows(\varphi_j, s) \wedge Apply(c, s, s')$ holds in M. Then, by definition of case plan, we can conclude that $\neg Knows(\varphi_i, s)$ holds in M, for $i \neq j$, $1 \leq i \leq l$. Hence, by Item (v) of Lemma 2.3 (for $c'' = []$, $\varphi = \varphi_j$, $c = p_j$) we have that

$\exists s''.Apply(p_j, s, s'') \wedge Apply([], s'', s')$ holds in M. (1)

From Item (i) of Lemma 2.3, we have that $s'' = s'$. Hence, (1) is equivalent to,

$Apply(p_j, s, s')$ holds in M. (2)

It follows from the assumption that $Knows(\varphi_j, s)$ holds in M and (2) that $Knows(\varphi_j, s) \wedge Apply(p_j, s, s')$ holds in M, which proves (i).

(ii) **Right to Left:** Assume that $Knows(\varphi_j, s) \wedge Apply(p_j, s, s')$ holds in M. Similar argument as above concludes that $\neg Knows(\varphi_i, s)$ holds in M, for $i \neq j$, $1 \leq i \leq l$. Hence, by definition of $Apply$ (case 6, for $r'' = []$, r' is the sequences $[(\varphi_1, p_1), \ldots, (\varphi_{j-1}, p_{j-1}), (\varphi_{j+1}, p_{j+1}), \ldots, (\varphi_l, p_l)]$), we have that

$Knows(\varphi_j, s) \wedge Apply(p_j, s, s') \wedge Apply([], s', s'') \supset Apply([case([(\varphi_j, p_j)|r'])], s, s'')$ holds in M.

Furthermore, from (i) of Lemma 2.3, we have that $s'' = s'$. Hence, we conclude that $Apply(c, s, s') \wedge Knows(\varphi_j, s)$ holds in M. This proves (ii).

The lemma follows from (i) and (ii). □

Lemma 2.6 *Assume that* $c = c_1, \ldots, c_n$ *is a conditional plan where* $c_1, \ldots, c_n$ *is a sequence of conditional plans satisfying the conditions of Observation 3.1. Let* c_1 *be a case plan of the form*

Case

$\varphi_1 \to p_1$

$\vdots$

$\varphi_l \to p_l$

Endcase

and s *be a situation term. Let* M *be a model of* B_{Apply} *such that* $M \models Knows(\varphi_j, s)$ *for some* j, $1 \leq j \leq l$. *Then,* $M \cup B_{\text{Apply}} \models Apply(c, s, s') \equiv Apply(c^*, s, s')$ *where* $c^* = p_j; c_2; \ldots; c_n$.

Proof. By Lemma 2.4, there exists $s_1, \ldots, s_{n-1}$ such that $Apply(c, s, s') \equiv Apply(c_1, s, s_1) \wedge \ldots \wedge Apply(c_n, s_{n-1}, s')$ holds in $M \cup B_{\text{Apply}}$ (1)

Since $M \models Knows(\varphi_j, s)$, by Lemma 2.5 and from (1), we have that

$Knows(\varphi_j, s) \wedge Apply(c_1, s, s_1) \equiv Knows(\varphi_j, s) \wedge Apply(p_j, s, s_1)$ holds in $M \cup B_{\text{Apply}}$ (2)

It follows from (1) and (2) that

$M \cup B_{\text{Apply}} \models Knows(\varphi_j, s) \wedge Apply(c, s, s') \equiv Knows(\varphi_j, s) \wedge Apply(p_j, s, s_1) \wedge Apply(c_2, s_1, s_3) \wedge \ldots \wedge Apply(c_n, s_{n-1}, s')$

which implies that

$M \cup B_{\text{Apply}} \models Apply(c, s, s') \equiv Apply(c^*, s, s')$ (By Lemma 2.4) (3)

The lemma follows from (3). □

Lemma 2.7 *Assume that $c = c_1, \ldots, c_n$ is a conditional plan where $c_1, \ldots, c_n$ is a sequence of conditional plans satisfying the conditions of Observation 3.1. Let c_1 be a sequence of actions and c_2 be a case plan of the form*

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

and s be a situation term. Let M be a model of B_{Apply} such that $M \models Knows(\varphi_j, do(c_1, s))$ for some j, $1 \leq j \leq l$. Then, $M \cup B_{\text{Apply}} \models Apply(c, s, s') \equiv Apply(c^*, s, s')$ where $c^* = c_1; p_j; \ldots; c_n$.

Proof. By Lemma 2.4, there exists $s_1, \ldots, s_{n-1}$ such that $Apply(c, s, s') \equiv Apply(c_1, s, s_1) \wedge \ldots \wedge Apply(c_n, s_{n-1}, s')$ holds in $M \cup B_{\text{Apply}}$ (1)

Since c_1 is a sequence of actions, we have that $s_1 = do(c_1, s)$. Therefore, from $M \models Knows(\varphi_j, do(c_1, s))$, $s_1 = do(c_1, s)$, and by Lemma 2.5, we have that

$M \cup B_{\text{Apply}} \models Apply(c_2, s_1, s_2) \equiv Apply(p_j, s_1, s_2)$. (2)

It follows from (1) and (2) that

$M \cup B_{\text{Apply}} \models Apply(c, s, s') \equiv Apply(c_1, s, s_1) \wedge Apply(p_j, s_1, s_2) \wedge Apply(c_3, s_1, s_3) \wedge \ldots \wedge Apply(c_n, s_{n-1}, s')$

which implies that

$M \cup B_{\text{Apply}} \models Apply(c, s, s') \equiv Apply(c^*, s, s')$ (By Lemma 2.4) (3)

The lemma follows from (3). □

Lemma 2.8 *Let D be a domain description and $M = (\sigma_0, \Phi)$ be a model of D. Then, there exists a model M_R of $R(D)$ such that for any fluent F and conditional plan c,*

1. *F is known to be true in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$;*
2. *F is known to be false in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(\neg F, s) \wedge Apply(c, S_0, s)$;*
3. *$\hat{\Phi}(c, \sigma_0) = \perp$ iff $Apply(c, S_0, \perp)$ holds in $M_R \cup B_{\text{Apply}}$.*

Proof. From the observation 3.1, we can assume that $c = c_1; \ldots; c_n$ where c_i is a sequence of actions or a case plan and for every i, $1 \leq i \leq n-1$, if c_i is a sequence of actions then c_{i+1} is a case plan.

Let M_R be the model of $R(D)$ constructed as in Lemma 2.1. We will prove by induction over the number of case plan in c, $count(c)$, that M_R satisfies the lemma .

Base case: $count(c) = 0$. By definition of $Apply$, and from (iv) of Lemma 2.3, we have that $Apply(c, S_0, s) \supset s = do(c, S_0)$. (1)

By Lemma 2.1, we have that F is known to be true in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \models Knows(F, do(c, S_0))$. (2)

It follows from (1) and (2) that

F is known to be true in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$. (3)

We can conclude the base case for the first item of the lemma. Similarly, we can prove the second item of the lemma. Since c is a sequence of actions, $\hat{\Phi}(c, \sigma_0) \neq \perp$, and by Item (vi) of Lemma 2.3, $Apply(c, S_0, s) \supset s = do(c, S_0)$ hold in $M_R \cup B_{\text{Apply}}$. This, together with Item (vii) of Lemma 2.3, proves that $Apply(c, S_0, \perp)$ does not hold in $M_R \cup B_{\text{Apply}}$. Hence, the base case for the third item of the lemma is proved. This, together with (2) and (3) proves the base case of the lemma.

Inductive step: Assume that we have proved the lemma for $count(c) \leq k$. We need to prove the lemma for $count(c) = k + 1$.

(I) F is known to be true in $\hat{\Phi}(c, \sigma_0)$. We will show that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

We consider two cases:

(1a) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$ we have that $\perp \neq \hat{\Phi}(c, \sigma_0)$. It implies that there exists j, $1 \leq j \leq l$, such that φ_j holds in σ_0. (4)

Let $c^* = p_j; c_2; \ldots; c_n$. Then, by definition of $\hat{\Phi}$ and from (4) we have that

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (5)

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$, from (5), we can conclude that F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$. (6)

It follows from (4) and Lemma 2.1 that $Knows(\varphi_j, S_0)$ holds in M_R. (7)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis, we can conclude that

$Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (8)

It follows from Lemma 2.6 that $Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (9)

From (8) and (9) we have that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

(1b) c_1 is a sequence of actions. Then, c_2 is a case plan. Let us assume that c_2 is the case plan.

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$ we have that $\bot \neq \hat{\Phi}(c, \sigma_0)$. This implies that there exists j, $1 \leq j \leq l$, such that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. (10)

Let $c^* = c_1; p_j; c_3; \ldots; c_n$. From (10) and the definition of $\hat{\Phi}$, we have that $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. This implies that F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$. (11)

From Lemma 2.7, we have that

$Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (12)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (11), we conclude that

$Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. This, together with (12), proves that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

From the two cases (1a) and (1b), we can conclude that if F is known to be true in $\hat{\Phi}(c, \sigma_0)$ then $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

(II) $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. We will prove that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

We consider two cases:

(2a) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$, we can conclude that there exists j, $1 \leq j \leq l$, such that $Knows(\varphi_j, S_0)$ holds in M_R. (13)

Let $c^* = p_j; c_2; \ldots; c_n$. By Lemma 2.6 we have that

$Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (14)

This implies that $Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (15)

Furthermore, from the definition of M_R and (13), we have that φ_j holds in σ_0. This implies that

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (16)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (15), we can conclude that

F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$, and from (16), we have that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

(2b) c_1 is a sequence of actions. Then, c_2 is a case plan. Let us assume that c_2 is the case plan.

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Similar to the case (2a), we conclude that there exists j, $1 \leq j \leq l$, such that
$Knows(\varphi_j, s') \wedge Apply(c_1, S_0, s')$ holds in $M_R \cup B_{\text{Apply}}$. (17)

Let $c^* = c_1; p_j; c_3 \ldots ; c_n$. From (17) and Lemma 2.7, we have that $Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (18)

By inductive hypothesis, we have that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. Hence,
$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (19)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis, we can conclude that

F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$. This, together with (19), proves that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

From the two cases (2a) and (2b), we can conclude that if $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$ then F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

The two cases (I) and (II) show that F is known to be true in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$. (20)

Similarly, we can show that F is known to be false in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(\neg F, s) \wedge Apply(c, S_0, s)$. (21)

The proof of the last item of the lemma, $\hat{\Phi}(c, \sigma_0) = \perp$ iff $Apply(c, S_0, \perp)$ holds in $M_R \cup B_{\text{Apply}}$ has also four cases similar to the cases (1a)-(1b) and (2a)-(2b). We will show next the first case. The other cases are similar.

Assume that $\hat{\Phi}(c, \sigma_0) = \perp$ where $c = c_1, \ldots, c_n$ and c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

We will show that $Apply(c, S_0, \perp)$ holds in $M_R \cup B_{\text{Apply}}$. We consider two cases:

1. there exists no j such that φ_j holds in σ_0 in M_R. Applies the last definition of *Apply* l times, we have that

   $Apply(c, S_0, S') \supset Apply([case([])|c'], S_0, S')$ holds in B_{Apply} where $c' = c_2, \ldots, c_n$. By the third item in the definition, we can then conclude that $Apply(c, S_0, S') \supset Apply(c, S_0, \bot)$ holds in $M_R \cup B_{\text{Apply}}$, i.e., the inductive step is proved.

2. there exists some j such that φ_j holds in σ_0 in M_R. Then, by Lemma 2.6, we have that

   $Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

   where $c^* = p_j; c_2; \ldots; c_n$. This, together with the fact that $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0) = \bot$ and the inductive hypothesis implies that $Apply(c, S_0, \bot)$ holds in B_{Apply}

The above two cases prove the inductive step for item 3 of the lemma. (22)

The lemma follows from (20), (21), and (22). □

Lemma 2.9 *Let D be a domain description and M_R be a model of $R(D)$. Then, there exists a model $M = (\sigma_0, \Phi)$ of D such that for any fluent F and any conditional plan c,*

1. *$M_R \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s))$ iff F is known to be true in $\hat{\Phi}(c, \sigma_0)$;*

2. *$M_R \cup B_{\text{Apply}} \models Knows(\neg F, s) \wedge Apply(c, S_0, s))$ iff F is known to be false in $\hat{\Phi}(c, \sigma_0)$;*

3. *$\hat{\Phi}(c, \sigma_0) = \bot$ iff $Apply(c, S_0, \bot)$ holds in $M_R \cup B_{\text{Apply}}$.*

Proof. From the observation 3.1, we can assume that $c = c_1; \ldots; c_n$ where c_i is a sequence of actions or a case plan and for every i, $1 \leq i \leq n-1$, if c_i is a sequence of actions then c_{i+1} is a case plan.

Let $M = (\sigma_0, \Phi)$ be the model of D constructed as in Lemma 2.2. We will prove by induction over the number of case plan in c, $count(c)$, that M satisfies the lemma.

Base case: $count(c) = 0$. By Item (vi) of the Lemma 2.3, we have that $B_{\text{Apply}} \models Apply(c, S_0, s) \equiv s = do(c, S_0)$. (1)

By Lemma 2.2, we have that

$M_R \models Knows(F, do(c, S_0))$ iff F is known to be true in $\hat{\Phi}(c, \sigma_0)$. This, together with (1), proves that

$M_R \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$ iff F is known to be true in $\hat{\Phi}(c, \sigma_0)$. (2)

Similarly, we have that

$M_R \cup B_{\text{Apply}} \models Knows(\neg F, s) \wedge Apply(c, S_0, s)$ iff F is known to be false in $\hat{\Phi}(c, \sigma_0)$. (3)

Furthermore, since c is a sequence of actions, $\hat{\Phi}(c, \sigma_0) \neq \bot$, and by Item (vi) Lemma 2.3, $Apply(c, S_0, s) \supset s = do(c, S_0)$ holds in $M_R \cup B_{\text{Apply}}$ This, together with Item (vii) of Lemma 2.3, proves that $Apply(c, S_0, \bot)$ does not hold in $M_R \cup B_{\text{Apply}}$. Hence, the base case for the third item of the lemma is proved. (4)

The base case of the lemma follows from (2)-(4).

Inductive step: Assume that we have proved the lemma for $count(c) \leq k$. We need to prove the lemma for $count(c) = k+1$.

(I) F is known to be true in $\hat{\Phi}(c, \sigma_0)$. We will show that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

We consider two cases:

(1a) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$ we have that $\perp \neq \hat{\Phi}(c, \sigma_0)$. This implies that there exists j, $1 \leq j \leq l$, such that φ_j holds in σ_0. (5)

Let $c^* = p_j; c_2; \ldots; c_n$. Then, by definition of $\hat{\Phi}$ and from (1) we have that

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (6)

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$, from (6), we can conclude that F is known to be true in$\hat{\Phi}(c^*, \sigma_0)$. (7)

By Lemma 2.6, we have that

$Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (8)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (7), we can conclude that

$Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. This, together with (8), proves that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

(1b) c_1 is a sequence of actions. Then, c_2 is a case plan. Again, let us assume that c_2 is the case plan.

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since F is known to be true in $\hat{\Phi}(c, \sigma_0)$ we have that $\perp \neq \hat{\Phi}(c, \sigma_0)$. It implies that there exists j, $1 \leq j \leq l$, such that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. (9)

Let $c^* = c_1; p_j; c_3; \ldots; c_n$. From (9), we have that $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (10)

It follows from (10) and Lemmas 2.7 that

$Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (11)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (10), we can conclude that

$Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. This, together with (11), proves that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

From the two cases (1a) and (1b), we can conclude that if F is known to be true in $\hat{\Phi}(c, \sigma_0)$ then $Knows(f, s) \wedge Apply(c, s_0, s)$ holds in $M_R \cup B_{\text{Apply}}$.

(II) $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. We will prove that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

We consider two cases:

(2a) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Since $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$, by Items (iii)-(v) of Lemma (2.4), we can conclude that there exists j, $1 \leq j \leq l$, such that $Knows(\varphi_j, S_0)$ holds in M_R. (12)

Let $c^* = p_j; c_2; \ldots; c_n$. Then, by Lemma 2.6, we conclude that

$Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (13)

From our assumption that $Knows(F, s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$, we have that $Knows(F, s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (14)

Furthermore, from the definition of M and (12), we have that φ_j holds in σ_0. This implies that

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (15)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (14), we can conclude that

F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$, and from (15), we can conclude that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

(2b) c_1 is a sequence of actions. Then, c_2 is a case plan. Again, let us assume that c_2 is the case plan.

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Similar to the case (2a), we conclude that there exists j, $1 \leq j \leq l$, such that $Knows(\varphi_j, s') \wedge Apply(c_1, S_0, s')$ holds in $M_R \cup B_{\text{Apply}}$. (16)

Let $c^* = c_1; p_j; c_3; \ldots; c_n$. From (16) and Lemmas 2.7, we have that $Apply(c, S_0, s) \equiv Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (17)

By (17) and our assumption, we have that

$Knows(F,s) \wedge Apply(c^*, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$. (18)

Since c_1 is a sequence of actions, we have that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. Hence, $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (19)

Since $count(c^*) \leq count(c) - 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis and (18), we conclude that

F is known to be true in $\hat{\Phi}(c^*, \sigma_0)$. This, together with (19), proves that F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

From the two cases (2a) and (2b), we conclude that if $Knows(F,s) \wedge Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$ then F is known to be true in $\hat{\Phi}(c, \sigma_0)$.

The two cases (I) and (II) show that F is known to be true in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(F,s) \wedge Apply(c, S_0, s)$. (20)

Similarly, we can show that F is known to be false in $\hat{\Phi}(c, \sigma_0)$ iff $M_R \cup B_{\text{Apply}} \models Knows(\neg F, s) \wedge Apply(c, S_0, s)$. (21)

The proof of the third item of the lemma, i.e., $\hat{\Phi}(c, \sigma_0) = \perp$ iff $Apply(c, S_0, \perp)$ holds in $M_R \cup B_{\text{Apply}}$ is similar to the proof of the item 3 of Lemma 2.8 and is omitted here. This, together with (20) and (21) proves the inductive step of the lemma, and hence, proves the lemma. □

Proposition 2 Let D be a domain description and $R(D)$ be the corresponding first order theory. Let c be a conditional plan and φ be a fluent formula. Then,

$$D \models \varphi \textbf{ after } c \quad \text{iff} \quad R(D) \cup B_{\text{Apply}} \models Apply(c, S_0, s) \wedge Knows(\varphi, s).$$

Proof. It suffices to prove the proposition for the case φ is a fluent literal F.

1. Assume that $D \models_{\mathcal{A}_K} F$ **after** c. We will prove that $R(D) \cup B_{\text{Apply}} \models Knows(F,s) \wedge Apply(c, S_0, s)$. Assume the contrary, $R(D) \cup B_{\text{Apply}} \not\models Knows(F,s) \wedge Apply(c, S_0, s)$. By definition, there exists a model M_R of $R(D)$ such that $M_R \cup B_{\text{Apply}} \not\models Knows(F,s) \wedge Apply(c, S_0, s)$. There are two possibilities

   (a) $Apply(c, S_0, \perp)$ holds in $M_R \cup B_{\text{Apply}}$. Hence, by Lemma 2.9, there exists a model $M = (\sigma_0, \Phi)$ of D such that $\hat{\Phi}(c, \sigma_0) = \perp$. This implies that $D \not\models_{\mathcal{A}_K} F$ **after** c. Hence, this case cannot happen. (1)

   (b) $Apply(c, S_0, s)$ holds in $M_R \cup B_{\text{Apply}}$ and $M_R \cup B_{\text{Apply}} \not\models Knows(F, s)$. Then, by Lemma 2.9, there exists a model M of D such that $M \not\models F$ **after** c. This implies that $D \not\models F$ **after** c. This contradicts with the assumption that $D \models F$ **after** c. Hence, this case cannot happen too. (2)

   From (1) and (2), we conclude that our assumption is incorrect, i.e., we have proved that $R(D) \cup B_{\text{Apply}} \models Knows(F,s) \wedge Apply(c, S_0, s)$. Therefore, we have that

   if $D \models_{\mathcal{A}_K} F$ **after** c then $R(D) \cup B_{\text{Apply}} \models Knows(F,s) \wedge Apply(c, S_0, s)$. (3)

2. Assume that $R(D) \cup B_{\text{Apply}} \models Knows(F,s) \wedge Apply(c, S_0, s)$. We will prove that $D \models_{\mathcal{A}_K} F$ **after** c. Assume the contrary, $D \not\models_{\mathcal{A}_K} F$ **after** c. This means that there exists a model $M = (\sigma_0, \Phi)$ of D such that $M \not\models F$ **after** c. There are two sub-cases:

(a) $\hat{\Phi}(c, \sigma_0) = \perp$. Then, by Lemma 2.8, there exists a model M_R of $R(D)$ such that $M_R \cup B_{\text{Apply}} \models Apply(c, S_0, \perp)$. This implies that $M_R \cup B_{\text{Apply}} \not\models Knows(F, s) \wedge Apply(c, S_0, s)$, which contradicts with our assumption. Therefore, this case cannot happen. (4)

(b) $\hat{\Phi}(c, \sigma_0) \neq \perp$. Then, F is not known to be true in $\hat{\Phi}(c, \sigma_0)$. Then, by Lemma 2.8, there exists a model M_R of $R(D)$ such that $M_R \cup B_{\text{Apply}} \not\models Knows(F, s) \wedge Apply(c, S_0, s)$. This implies that $R(D) \cup B_{\text{Apply}} \not\models Knows(F, s) \wedge Apply(c, S_0, s)$. This contradicts our assumption that $R(D) \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$. Hence, this case cannot happen too. (5)

From (4) and (5, we have that $D \models_{\mathcal{A}_K} F$ **after** c. Hence, we have that

$$\text{if } R(D) \models Knows(F, s) \wedge Apply(c, S_0, s) \text{ then } D \models_{\mathcal{A}_K} F \textbf{ after } c. \quad (6)$$

From (3) and (6), we can conclude that
$D \models_{\mathcal{A}_K} F$ **after** c iff $R(D) \cup B_{\text{Apply}} \models Knows(F, s) \wedge Apply(c, S_0, s)$. □

Appendix B

In this section we prove the soundness of the ω-Approximation w.r.t. the semantics of $\mathcal{A}_K$. We will need the following notations and lemmas.

Let $\sigma = \langle T, F \rangle$ be an a-state and $\delta = \langle s, \Sigma \rangle$ be a k-state. We say σ *agrees with* δ if for every state $s \in \Sigma$, $T \subseteq s$ and $F \cap s = \emptyset$.

For an a-state $\sigma = \langle T, F \rangle$, by $true(\sigma)$ and $false(\sigma)$ we denote the set T and F respectively.

In this section, by Φ and Φ_ω we denote the transition function of D w.r.t. the semantics of $\mathcal{A}_K$ and the ω-Approximation respectively. Res and Res_ω denote the result functions of D w.r.t. the $\mathcal{A}_K$ semantics and the ω-Approximation respectively.

For a sequence of actions $\alpha = a_1, \ldots, a_n$ $(n \geq 1)$ and a state s, by $Res(\alpha, s)$ we denote the state $Res(a_n, \ldots, Res(a_1, s))$. Similarly $Res_\omega(\alpha, \sigma)$ denotes the state $Res_\omega(a_n, \ldots, Res_\omega(a_1, \sigma))$.

The following observations are trivial and will be used in the proofs in this section.

Observation 3.1 *Let $\sigma = \langle T, F \rangle$ be an a-state and $\delta = \langle s, \Sigma \rangle$ be a grounded k-state such that σ agrees with δ, then*

1. *if φ is a conjunction of literals and φ holds in σ ($\sigma \models \varphi$), then for every $s \in \Sigma$, φ holds in s;*
2. *for every action a, $\Phi(a, \delta)$ is a grounded k-state;*
3. *$\Sigma \subseteq Comp(\sigma)$[5];*
4. *if α is a sequence of non-sensing actions and σ is complete,*

$$Res_\omega(\alpha, \sigma) = Res(\alpha, T).$$

The proof of Proposition 6 is based on the following lemmas.

[5] Recall that we simply write T instead of $\langle T, F \rangle$ when $\langle T, F \rangle$ is a complete a-state.

Lemma 3.10 *Let D be a domain description, σ be an a-state, and δ be a grounded k-state of D such that σ agrees with δ. Then, for every sequence of non-sensing actions α of D, $\mathcal{R}es(\alpha, \sigma)$ agrees with $\hat{\Phi}(\alpha, \delta)$.*

Proof. Assume that $\sigma = \langle T, F \rangle$, $\delta = \langle s, \Sigma \rangle$, and $\hat{\Phi}(\alpha, \delta) = \langle \hat{s}, \hat{\Sigma} \rangle$.

Let $f \in true(\mathcal{R}es(\alpha, \sigma))$.
$\Rightarrow f \in \bigcap_{\sigma' \in Comp(\sigma)} true(Res_\omega(\alpha, \sigma'))$. (by definition of $\mathcal{R}es$)
$\Rightarrow \forall \sigma' \in Comp(\sigma) \quad f \in true(Res_\omega(\alpha, \sigma'))$.
$\Rightarrow \forall \sigma' \in Comp(\sigma) \quad f \in Res(\alpha, true(\sigma'))$. (by Item 4, Observation 3.1)
$\Rightarrow \forall s' \in \Sigma \quad f \in Res(\alpha, s')$. (by Item 3, Observation 3.1)
$\Rightarrow \forall s^* \in \hat{\Sigma} \quad f \in s^*$. (*)

Let $f \in false(\mathcal{R}es(\alpha, \sigma))$.
$\Rightarrow f \in \bigcap_{\sigma' \in Comp(\sigma)} false(Res_\omega(\alpha, \sigma'))$. (by definition of $\mathcal{R}es$)
$\Rightarrow \forall \sigma' \in Comp(\sigma) \quad f \in false(Res_\omega(\alpha, \sigma'))$.
$\Rightarrow \forall \sigma' \in Comp(\sigma) \quad f \notin Res(\alpha, true(\sigma'))$. (by Item 4, Observation 3.1)
$\Rightarrow \forall s \in \Sigma \quad f \notin Res(\alpha, s)$. (by Item 3, Observation 3.1)
$\Rightarrow \forall s^* \in \hat{\Sigma} \quad f \notin s^*$. (**)

The lemma follows from (*) and (**). □

Lemma 3.11 *Let D be a domain description, σ be an a-state, and δ be a grounded k-state of D such that σ agrees with δ. Then, for every sensing action a of D, there exists $\sigma' \in \Phi_\omega(a, \sigma)$ such that σ' agrees with $\Phi(a, \delta)$.*

Proof. Assume that a occurs in the following k-propositions: a **determines** $f_1, \ldots,$ a **determines** f_n
By definition, we have that $\mathcal{K}(a, \sigma) = \{f_1, \ldots, f_n\}$.

Assume that $\sigma = \langle T, F \rangle$ and $\delta = \langle s, \Sigma \rangle$.

Let $K_1 = s \cap \mathcal{K}(a, \sigma)$ and $K_2 = \mathcal{K}(a, \sigma) \setminus s$.

Since δ is a grounded k-state, we have that $s \in \Sigma$. From the assumption that σ agrees with δ, we have that $T \subseteq s$ and $F \cap s = \emptyset$. This, together with the definitions of K_1 and K_2, implies that $K_1 \cap F = \emptyset$ and $K_2 \cap T = \emptyset$. Therefore, we have that
$\sigma' = \langle T \cup K_1, F \cup K_2 \rangle \in \Phi_\omega(a, \sigma)$. (1)

Let $\Phi(a, \delta) = \langle s, \Sigma' \rangle = \delta'$. (2)

Consider any s' from Σ'. By definition of $\Phi(a, \delta)$, we have that
$s' \cap \{f_1, \ldots, f_n\} = s \cap \{f_1, \ldots, f_n\} = K_1$ and $\{f_1, \ldots, f_n\} \setminus s' = \{f_1, \ldots, f_n\} \setminus s = K_2$. Thus, $K_1 \subseteq s'$ and $s' \cap K_2 = \emptyset$. Since σ agrees with δ we have that $T \subseteq s'$ and $F \cap s' = \emptyset$. Therefore, $T \cup K_1 \subseteq s'$ and $(F \cup K_2) \cap s' = \emptyset$. (3)

Since (3) holds for every state $s' \in \Sigma'$, we have that $\langle T \cup K_1, F \cup K_2 \rangle$ agrees with δ'. (4)

The lemma follows from (1) and (4). □

The next lemma is the generalization of the lemmas 3.10 and 3.11 to a sequence of both sensing and non-sensing actions.

Lemma 3.12 *Let D be a domain description, σ be an a-state, and δ be a grounded k-state of D such that σ agrees with δ. Then, for every sequence of actions α, there exists a state $\sigma' \in \Phi_\omega(\alpha, \sigma)$ such that σ' agrees with $\hat{\Phi}(\alpha, \delta)$.*

Proof. Let $n_s(\alpha)$ be the number of sensing actions occurring in α. We prove the lemma by induction over $n_s(\alpha)$.

Base case: $n_s(\alpha) = 0$, i.e., α is a sequence of non-sensing actions. By Lemma 3.10, we have that $\mathcal{R}es(\alpha, \sigma)$ agrees with $\hat{\Phi}(\alpha, \delta)$. Since $\alpha = pre(\alpha)$, by definition of Φ_ω, we have that $\Phi_\omega(\alpha, \sigma) = \{\mathcal{R}es(\alpha, \sigma)\}$. This proves the base case.

Inductive step: Assume that the first sensing action occurring in α is a, i.e., $\alpha = \beta; a; \gamma$ where β does not contain a sensing action. Let $\hat{\Phi}(\beta, \delta) = \delta_1$ and $\mathcal{R}es(\beta, \sigma) = \sigma_1$. Then, by Lemma 3.10, we have that

σ_1 agrees with δ_1. (a)

Since δ is a grounded k-state and β is a sequence of non-sensing actions, using Observation 3.1, we can easily prove that δ_1 is a grounded k-state. Hence, by Lemma 3.11,

$\exists \sigma_2 \in \Phi_\omega(a, \sigma_1)$ such that σ_2 agrees with $\Phi(a, \delta_1) = \delta_2$. (b)

Since $n_s(\gamma) = n_s(\alpha) - 1$, by the induction hypothesis, we conclude that

$\exists \sigma_3 \in \Phi_\omega(\gamma, \sigma_2)$ such that σ_3 agrees with $\hat{\Phi}(\gamma, \delta_2) = \delta_3$. (c)

From $\mathcal{R}es(\beta, \sigma) = \sigma_1$, $\sigma_2 \in \Phi_\omega(a, \sigma_1)$, $\sigma_3 \in \Phi_\omega(\gamma, \sigma_2)$, and by definition of Φ_ω, we have that $\sigma_3 \in \Phi_\omega(\alpha, \sigma)$ (d)

From $\hat{\Phi}(\beta, \delta) = \delta_1$, $\Phi(a, \delta_1) = \delta_2$, $\hat{\Phi}(\gamma, \delta_2) = \delta_3$, and by definition of $\hat{\Phi}$, we have that $\hat{\Phi}(\alpha, \delta) = \delta_3$. (e)

Since σ_3 agrees with δ_3, from (d) and (e), we can conclude the induction step. Hence, the lemma is proved. $\Box$

In the next lemma we extend the result of Lemma 3.12 to an arbitrary conditional plan.

Lemma 3.13 *Let D be a domain description, σ be an a-state, and δ be a grounded k-state of D such that σ agrees with δ. Then, for every conditional plan c where $\bot \not\in \Phi_\omega(c, \sigma)$, there exists an a-state $\sigma' \in \Phi_\omega(c, \sigma)$ such that σ' agrees with $\hat{\Phi}(c, \delta)$.*

Proof. By Observation 3.1, we know that c can be represented as a sequence of conditional plans $c_1; \ldots; c_n$ where c_i is either a sequence of actions or a case plan and for every $i < n$ if c_i is a sequence of actions then c_{i+1} is a case plan. We prove the lemma by induction over $count(c)$, the number of case plans in c.

The base case, $count(c) = 0$, is proved by Lemma 3.12.

We now prove the inductive step, i.e., assume that the lemma is shown for $count(c) \leq k$, we prove the lemma for $count(c) = k + 1$. We consider two cases:

(1) c_1 is a case plan. Assume that c_1 is the following plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

From the assumption that $\bot \notin \Phi_\omega(c, \sigma)$ (in the statement of the Lemma), we can conclude that there exists j, $1 \leq j \leq l$, such that φ_j holds in σ. Let $c^* = p_j; c_2 \ldots; c_n$. Then, by definition of Φ_ω we have that

$$\Phi_\omega(c, \sigma) = \Phi_\omega(c^*, \sigma). \tag{a1}$$

Since $count(c) > count(c^*) + 1$, we have that $count(c^*) \leq k$. It follows from the induction hypothesis that

there exists a state $\sigma' \in \Phi_\omega(c^*, \sigma)$ such that σ' agrees with $\hat{\Phi}(c^*, \delta)$. (a2)

Since σ agrees with δ and φ_j holds in σ, we can conclude that φ_j holds in δ, which implies that

$$\hat{\Phi}(c, \delta) = \hat{\Phi}(c^*, \delta). \tag{a3}$$

From (a3), (a2), and (a1), we have that $\sigma' \in \Phi_\omega(c, \sigma)$ and σ' agrees with $\hat{\Phi}(c, \delta)$. This proves the lemma for the case (1).

(2) c_1 is a sequence of actions. Let $c^* = c_2; \ldots; c_n$. Then, by definitions of $\hat{\Phi}$ and Φ_ω, we have that

$$\hat{\Phi}(c, \delta) = \hat{\Phi}(c^*, \hat{\Phi}(c_1, \delta)) \tag{b1}$$

and

$$\hat{\Phi}_\omega(c, \sigma) = \bigcup_{\sigma' \in \Phi_\omega(c_1, \sigma)} \hat{\Phi}(c^*, \sigma'). \tag{b2}$$

Since σ agrees with δ and δ is a grounded k-state, by Lemma 3.12, we know that

there exists a state $\sigma_1 \in \hat{\Phi}_\omega(c_1, \sigma)$ such that σ_1 agrees with $\hat{\Phi}(c_1, \delta)$. (b3)

Since c^* starts with a case plan and $count(c^*) = k + 1$, from the first case, we can conclude that

there exists a state $\sigma' \in \Phi_\omega(c^*, \sigma_1)$ such that σ' agrees with $\hat{\Phi}(c^*, \hat{\Phi}(c_1, \delta))$ (b4)

From (b2) and (b4), we have that $\sigma' \in \Phi_\omega(c, \sigma)$ and σ' agrees with $\hat{\Phi}(c^*, \hat{\Phi}(c_1, \delta)) = \hat{\Phi}(c, \delta)$. Hence the inductive step is proved for case (2).

The inductive step follows from the above two cases. □

We are now ready to prove the Proposition 6.

Proposition 6 (Soundness of ω-Approximation w.r.t. $\models_{\mathcal{A}_K}$) Let D be a domain description, φ be a fluent formula, and c be a conditional plan. Then,

$$\text{if } D \models_\omega \varphi \textbf{ after } c \text{ then } D \models_{\mathcal{A}_K} \varphi \textbf{ after } c.$$

Proof. It is easy to see that to prove this proposition it suffices to show that for every fluent literal f, if $D \models_\omega f$ **after** c then $D \models_{\mathcal{A}_K} f$ **after** c.

Let us consider an arbitrary fluent f such that $D \models_\omega f$ **after** c.

Let σ_0 be the initial a-state of D. Since $D \models_\omega f$ **after** c, by definition of $\models_\omega$, we have that

$\perp \notin \Phi_\omega(c, \sigma_0)$, (a1)

and

for every $\sigma' \in \Phi_\omega(c, \sigma_0)$, f is true in σ', i.e., $\forall \sigma' \in \Phi_\omega(c, \sigma_0) : f \in true(\sigma')$. (a2)

Now, let us consider an arbitrary model (δ_0, Φ) of D. It is easy to see that σ_0 agrees with δ_0. Hence, by Lemma 3.13, we conclude that

there exists a state $\sigma' \in \Phi_\omega(c, \sigma_0)$, such that σ' agrees with $\hat{\Phi}(c, \delta_0)$. (b1)
$\Rightarrow f$ is known to be true in $\hat{\Phi}(c, \delta_0)$. (from (a2) and (b1)) (b2)

Since (b2) holds for every model of D, we can conclude that
$D \models_{\mathcal{A}_K} f$ **after** c. (c)

Similarly, we can prove that if $D \models_\omega \neg f$ **after** c then $D \models_{\mathcal{A}_K} \neg f$ **after** c. (d)

The proposition follows from (c) and (d). □

Appendix C

In this section, we prove the consistency of the program $\pi(D)$ and its soundness and completeness w.r.t. the rational semantics of $\mathcal{A}_K$. Recall that $\pi(D)$ is a sorted-program. Therefore, in this section, we will assume that each rule of $\pi(D)$ stands for the corresponding set of ground rules whose variables are instantiated using the corresponding sorts, for example, the rules (π.4) stands for the set of ground rules of the form

$$Holds(f, res(a, s)) \leftarrow Holds(f, s), not\ ab(f, a, s).$$

where f is a fluent, a is an action, and s is a situation term of the form $res(a_n, \ldots, res(a_1, s'))$, where a_i is an action of D for $i = 1, \ldots, n$ and s' is s_0 or a state constant listed in the set of language dependent rules. In the following we often write α_i and $res([a_1, \ldots, a_i], s')$ to denote a sequence of i actions of D and $res(a_i, \ldots, res(a_1, s'))$, respectively. Throughout the section, we will make use of the following lemmas.

Lemma 2.14 (Marek and Subrahmanian, [MS89]) *For any answer set A of an extend logic program π:*

(a) for any rule of the form

$$l \leftarrow l_1, \ldots, l_n, not\ l_{n+1}, \ldots, not\ l_{n+m}$$

if $\{l_1, \ldots, l_n\} \subseteq A$ and $\{l_{n+1}, \ldots, l_{n+m}\} \cap A = \emptyset$ then $l \in A$;[6]

(b) if A is a consistent answer set of π and $l \in A$ then there exists a rule of the form

$$l \leftarrow l_1, \ldots, l_n, not\ l_{n+1}, \ldots, not\ l_{n+m}$$

such that $\{l_1, \ldots, l_n\} \subseteq A$ and $\{l_{n+1}, \ldots, l_{n+m}\} \cap A = \emptyset$. □

[6] Often, the rule $l \leftarrow l_1, \ldots, l_n, not\ l_{n+1}, \ldots, not\ l_{n+m}$ is said to be *applicable* in A.

Lemma 2.15 (Gelfond and Lifschitz, [GL91]) *Let π be an extended logic program. If π has a consistent answer set then every answer set of π is consistent.*

We will need the following notations and lemmas.

Let $\pi_1(D)$ be the program consisting of

- the domain dependent rules of $\pi(D)$, i.e., rules $(\pi.1)$-$(\pi.3)$;
- the language dependent rules of $\pi(D)$; and
- the rules of $\pi(D)$ whose head are literals or disjunction of literals formed by the predicate symbols *holds*, *ab*, and *holds_list*, i.e., rules $(\pi.4)$-$(\pi.8)$ and $(\pi.12$-$(\pi.14)$.

We prove two lemmas which are essential for proving the consistency of $\pi(D)$.

Lemma 2.16 Let D be a consistent domain description. Then, $\pi_1(D)$ is consistent.

Proof. To prove the consistency of $\pi_1(D)$ we will construct a splitting sequence of $\pi_1(D)$ and use the splitting sequence theorem of [LT94].

Let V be the set of literals of $\pi_1(D)$ which are formed by the predicate symbols *initially*, *causes*, and *determines*, S be the set of state constants, i.e., $S = \{s_0\} \cup \{s \mid state(s) \in \pi(D)\}$. We define, the sequence $U = \langle U_i \rangle_{i<\infty}$ as follows.

$$\begin{array}{rl} U_i = & V \cup \{Holds(f, res([a_1, \ldots, a_n], s)) \mid n \leq i, a'_k s \text{ are actions in } D, f \text{ is a fluent literal}, s \in S\} \cup \\ & \{ab(f, a, res([a_1, \ldots, a_n], s)) \mid 1 \leq n \leq i, a'_k s \text{ are actions in } D, f \text{ is a fluent literal}, s \in S\} \cup \\ & \{holds\_list([f_1, \ldots, f_m], res([a_1, \ldots, a_n], s)) \mid n \leq i, a'_k s \text{ are actions in } D, f'_j s \text{ are fluent literal}, s \in S\} \end{array}$$

The sequence $U = \langle U_i \rangle_{i<\infty}$ is monotone ($U_i \subset U_j$ whenever $i < j$) and continuous (for each limit ordinal $\alpha < \mu$, $U_\alpha = \bigcup_{\beta<\alpha} U_\beta$). It is easy to see that for each $i < \infty$, U_i is a splitting set of $\pi_1(D)$ and that $\bigcup_{i<\infty} U_i$ is equal to the set of all literals occurring in $\pi_1(D)$. Hence, U is a splitting sequence of $\pi_1(D)$. By the splitting sequence theorem existence of an answer set of $\pi_1(D)$ follows from existence of a solution to $\pi_1(D)$ (with respect to U). Let P_i be the collection of all the rules from $\pi_1(D)$ whose heads belong to U_i. To show existence of such a solution it suffices to

(i) show that P_0 has a consistent answer set X_0

(ii) assume that for i such that $i + 1 < \infty$ the program P_i has a consistent answer set X_i;

(iii) use this assumption to show that P_{i+1} also has a consistent answer set;

(iv) show that $\bigcup_{i<\infty} X_i$ is consistent.

Let us prove (i). We have that P_0 consists of rules of the form $(\pi.1)$-$(\pi.3)$, $(\pi.12)$-$(\pi.14)$, $(\pi.5)$-$(\pi.6)$, and the language dependent rules of $\pi(D)$. Let us denote the set language dependent rules of $\pi(D)$ by LD. We define

$I = \{initially(f) \mid$ **initially** $f \in D\}$,

$H_1 = \{Holds(F, s_0) \mid initially(F) \in I\}$,

$H_2 = \{holds(f, s_0) \mid holds(f, s_0) \notin H_1$ and $\neg holds(f, s_0) \notin H_1\}$, and

$H_3 = \{holds\_list([f_1, \ldots, f_m], s) \mid Holds(f_i, s) \in H_1 \cup H_2 \cup LD, \text{ for } i = 1, \ldots, m\}.$

It is easy to see that the set $X_0 = V \cup H_1 \cup H_2 \cup H_3 \cup LD$ is a consistent answer set of P_0, i.e., (i) is proved.

Let us now assume (ii) and show (iii). Let $P = e_{U_i}(P_{i+1} \setminus P_i, \cup_{j=0}^{i} X_j)$ be the result of partial evaluation of the program $P_{i+1} \setminus P_i$ with respect to the set $\cup_{j=0}^{i} X_j$. It is easy to check that P consists of rules of the following types

(a) $Holds(f, res([\alpha_i, a], s)) \leftarrow$

if a1. there exists an ef-proposition of the form a **causes** f **if** $p_1, \ldots, p_m$ such that

a2. $Holds(p_j, res(\alpha_i, s)) \in \cup_{j=0}^{i} X_j$, for $j = 1, \ldots, m$.

(b) $Holds(f, res([\alpha_i, a], s)) \leftarrow$

if b1. $Holds(f, res(\alpha_i, s)) \in \cup_{j=0}^{i} X_j$ and

b2. $ab(f, a, res(\alpha_i, s)) \notin \cup_{j=0}^{i} X_j$.

(c) $ab(\bar{f}, a, res(\alpha_i, s)) \leftarrow$

if c1. there exists an ef-proposition of the form a **causes** f **if** $p_1, \ldots, p_m$ such that

c2. $Holds(p_j, res(\alpha_i, s)) \in \cup_{j=0}^{i} X_j$, for $j = 1, \ldots, m$.

(d) $holds\_list([h|t], res(\alpha_i, s)) \leftarrow Holds(h, res(\alpha_i, s)), holds\_list(t, res(\alpha_i, s))$.

Let us denote the program consisting the rules (a), (b), and (c) of P by Q. We first prove that Q is consistent. Since Q contains only facts, to prove consistency of Q, it suffices to show that Q does not contain contradictory facts.

Assume the contrary, i.e., Q is inconsistent. Since Q does not contain negative literal of the form $ab(f, a, s)$, this is possible only if there exists a fluent f, a sequence of actions α_i, an action a, and a state constant s such that Q contains both $holds(f, res([\alpha_i, a], s))$ and $\neg holds(f, res([\alpha_i, a], s))$.

Consider the case $holds(f, res([\alpha_i, a], s))$ is a rule of type (a) of Q. Then, from (a1)-(a2), and (c1)-(c2), we can conclude that $ab(\bar{f}, a, res(\alpha_i, s)) \in P$ which implies that $\neg holds(f, res([\alpha_i, a], s))$ is not a rule of type (b) of Q, i.e., $\neg holds(f, res([\alpha_i, a], s))$ is a rule of type (a) of Q. It implies that there are two ef-propositions a **causes** f **if** $p_1, \ldots, p_m$ and a **causes** $\neg f$ **if** $q_1, \ldots, q_k$ and a situation s^* (which is $res(\alpha_i, s)$) and $\{p_1, \ldots, p_m\}$ and $\{q_1, \ldots, q_k\}$ are satisfied in s^*. This contradicts the assumption that D is consistent. Therefore, we conclude that $holds(f, res([\alpha_i, a], s))$ is a rule of type (b) of Q.

Similar argument shows that if $holds(f, res([\alpha_i, a], s))$ is a rule of type (b) of Q then $\neg holds(f, res([\alpha_i, a], s))$ is also a rule of type (b) of Q. Thus, by (b1) and (b2), we conclude that $holds(f, res(\alpha_i, s)) \in \cup_{j=0}^{i} X_j$ and $\neg holds(f, res(\alpha_i, s)) \in \cup_{j=0}^{i} X_j$ which contradicts the assumption that $\cup_{j=0}^{i} X_j$ is consistent. Hence, we conclude that Q is consistent, i.e., (iii) is proved.

It is easy to see that Q is a splitting set of P with $b_Q(P) = Q$, and the reduct of P w.r.t. Q contains only rules of the type

$holds\_list([h|t], res(\alpha_i, s)) \leftarrow holds\_list(t, res(\alpha_i, s)),$

which is a positive program, and hence, has a unique consistent answer set, say Y. By the splitting theorem, $X_{i+1} = Q \cup Y$ is a consistent answer set of P. This proves the consistency of P_{i+1}.

It is easy to see that $X_i \cap X_j = \emptyset$ for $i \neq j$, i.e., (iv) is proved.

From (i)-(iv) we conclude that there exists a consistent solution of $\pi_1(D)$ w.r.t. U, i.e. $\pi_1(D)$ is consistent. □

Lemma 2.17 $\pi(D)$ is consistent iff $\pi_1(D)$ is consistent.

Proof. It is easy to see that the set of literals U_1 formed by the predicate symbols belonging to $\pi_1(D)$ constitutes a splitting set of $\pi(D)$ and the bottom part of $\pi(D)$ with respect to U_1 is $\pi_1(D)$. It follows immediately from the splitting set theorem [LT94] that

(a) if $\pi(D)$ is consistent then $\pi_1(D)$ is consistent.

It remains to be shown that the reverse of (a) is also correct, i.e., assume that

(b) $\pi_1(D)$ is consistent,

we need to prove that $\pi(D)$ is consistent.

From (b), we can assume that $\pi_1(D)$ has a consistent answer set X_1.

Let us denote the reduct of $\pi(D)$ with respect to U_1 and X_1 by $P_1(D)$. By the splitting set theorem, the consistency of $\pi(D)$ is proved if $P_1(D)$ is consistent. Therefore, we will prove that $P_1(D)$ is consistent.

Let $U_2 = \{\neg agree(f, s_1, s_2) \mid \neg agree(f, s_1, s_2) \in lit(\pi(D))\}$. It is easy to see that U_2 is a splitting set of $P_1(D)$. The bottom program of $P_1(D)$ with respect to the splitting set U_2, $P_2(D) = b_{U_2}(P_1(D))$, consists of the following rules,

$$\begin{array}{ll} \neg agree(f, s_1, s_2). & \text{if } (holds(f, s_1) \in X_1, \neg holds(f, s_2) \in X_1) \\ & \text{or } (\neg holds(f, s_1) \in X_1, holds(f, s_2) \in X_1) \end{array} \tag{A.1}$$

$$\begin{array}{ll} \neg agree(f, s_2, s_1). & \text{if } (holds(f, s_1) \in X_1, \neg holds(f, s_2) \in X_1) \\ & \text{or } (\neg holds(f, s_1) \in X_1, holds(f, s_2) \in X_1) \end{array} \tag{A.2}$$

Since $P_2(D)$ consists of only facts which are negative literals of the form $agree(f, s, s')$, we have that $X_2 = P_2(D)$ is a consistent answer set of $P_2(D)$.

The reduct of $P_1(D)$ with respect to U_2 and X_2, $R = e_{U_2}(P_1(D) \setminus P_2(D), X_2)$, can then be splitted again by the set U_3 consisting of literals of $\pi(D)$ formed by the predicate symbol *accessible*. The bottom program of R with respect to U_3, denoted by $P_3(D)$, consists of the following rules:

$$\begin{array}{ll} \neg accessible(s, s_0). & \text{if } initially(f) \in X_1, \\ & \text{and } \neg holds(f, s) \in X_1, \\ & \text{and } state(s) \in X_1. \end{array} \tag{A.3}$$

$$\begin{array}{ll} \neg accessible(s, s_0). & \text{if } initially(\neg f) \in X_1, \end{array} \tag{A.4}$$

and $holds(f, s) \in X_1$,
and $state(s) \in X_1$.

$$accessible(s, s_0) \leftarrow not\ \neg accessible(s, s_0) \tag{A.5}$$

$$accessible(res(a, s_2), res(a, s_1)) \leftarrow accessible(s_2, s_1). \tag{A.6}$$
if there exists $causes(a, f, l) \in X_1$.

$$\neg accessible(res(s_g, s_2), res(s_g, s_1)) \leftarrow accessible(s_2, s_1). \tag{A.7}$$
if $determines(s_g, g) \in X_1$,
and $\neg agree(g, s_1, s_2) \in X_2$

$$accessible(res(s_g, s_2), res(s_g, s_1)) \leftarrow accessible(s_2, s_1),\ not\ \neg accessible(res(s_g, s_2), res(s_g, s_1)). \tag{A.8}$$
if $determines(s_g, g) \in X_1$

It is easy to see that the sequence $T = \langle T_i \rangle_{i<\infty}$ where $T_i = \{accessible(res(\alpha_j, s), res(\alpha_j, s')) \mid s, s'$ are state constants in $\pi_1(D)$ or s_0, α_j is a sequence of j actions of $D, 0 \leq j \leq i\} \cup \{accessible(res(\alpha_j, s), res(\alpha_j, s')) \mid s, s'$ are state constants in $\pi_1(D)$ or s_0, α_j is a sequence of j actions of $D, 0 \leq j \leq i\}$ is a splitting sequence of the program $P_3(D)$. (1)

Let Q_i be the program consisting of rules of $P_3(D)$ whose heads belong to T_i.

We have that Q_0 consists of only facts of the form $\neg accessible(s, s_0)$, and hence, has a unique consistent answer set, say Y_0 (here, we have that $Y_0 = Q_0$). (2)

The partial evaluation of the program $P_3(D)$ w.r.t. the set $\cup_{j=0}^{i} Y_j$, $e_{T_i}(Q_{i+1} \setminus Q_i, \cup_{j=0}^{i} Y_j)$ is a positive program. Hence, it has a consistent answer set too. (3)

It follows from (1)-(3) and the splitting sequence theorem that $P_3(D)$ has a consistent answer set. We denote it by X_3.

Finally, the reduct of $P_3(D)$ with respect to U_3 and X_3, denoted by $P_4(D)$, consists of the following rules

$$apply([], S, S). \tag{A.9}$$

$$apply([A|T], S, S1) \leftarrow apply(T, res(A, S), S1). \tag{A.10}$$
if $action(A) \in X_1$

$$\neg knows(F, S_1) \leftarrow \text{if } accessible(S_2, S_1) \in X_3, \tag{A.11}$$
and $Holds(\bar{F}, S_2) \in X_1$

$$knows(F, S) \leftarrow not\ \neg knows(F, S). \tag{A.12}$$

$$kwhether(F, S) \leftarrow knows(F, S). \tag{A.13}$$

$$kwhether(F, S) \leftarrow knows(\bar{F}, S). \tag{A.14}$$

$$knows\_after(F, C, S) \leftarrow apply(C, S, S1), S1 \neq \perp, knows(F, S1). \tag{A.15}$$

$$knows\_list([], S). \tag{A.16}$$

$$knows\_list([H|T], S) \leftarrow knows(H, S), knows\_list(T, S). \tag{A.17}$$

$$apply(X, \perp, \perp). \tag{A.18}$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow knows\_list(C, S), apply(P, S, S1), apply(L, S1, X). \tag{A.19}$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow not\ knows\_list(C, S), apply([case(T)|L], S, X). \tag{A.20}$$

$$apply([case([])|L], S, \perp). \tag{A.21}$$

Let Z be the set of negative literals of the form $\neg knows(f, s)$ occuring in the heads of rules of $P_4(D)$. Since negative literals of the form $\neg knows(f, s)$ occur only in the heads of rules whose body is empty, Z is a splitting set of $P_4(D)$. It is easy to see that the bottom $b_Z(P_4(D))$ has Z as its unique answer set. The partial evaluation of $P_4(D)$ with respect to Z is a positive program, and hence, has a unique answer set W. Thus, by the splitting theorem, $X_4 = Z \cup W$ be a consistent answer set of $P_4(D)$.

It follows from the splitting set theorem and the fact that X_i's are consistent and pairwise disjoint that $X_1 \cup X_2 \cup X_3 \cup X_4$ is a consistent answer set of $\pi(D)$, i.e., (b) is proved.

The lemma is proved by (a) and (b). □

We now prove the consistency of $\pi(D)$

Proposition 9 (Consistency of $\pi(D)$) For every consistent D, the program $\pi(D)$ is consistent.

Proof. Since D is consistent, by Lemma 2.16 we have that $\pi_1(D)$ is consistent. The consistence of $\pi(D)$ follows from Lemma 2.17. □

We now introduce some notations which will be used in proving Proposition 3. Let s be a situation term of $\pi(D)$, i.e., $s = res([a_1, \ldots, a_n], s')$ where s' is s_0 or a state constant occuring in the language dependent rules of $\pi(D)$. For a set of literals X of $\pi(D)$, we define

$$\sigma_{s,X} = \{f \mid holds(f, s) \in X\}$$

and

$$\Sigma_{s,X} = \{\sigma_{s',X} \mid accessible(s', s) \in X\}.$$

The following observation follows directly from the definition of $\sigma_{s,X}$.

Observation 2.2 *Let X and Y be consistent sets of literals of $\pi(D)$ and s be a situation term in $\pi(D)$ such that $X \subseteq Y$ and there is no literal of the form $Holds(f,s) \in Y \setminus X$. Then, $\sigma_{s,X} = \sigma_{s,Y}$.* $\Box$

Lemma 2.18 *Let D be a consistent domain description. Then,*

1. *for any rational model (σ_0, Φ) of D, there exists an answer set A of $\pi(D)$ such that $\sigma_0 = \langle \sigma_{s_0,A}, \Sigma_{s_0,A}\rangle$.*
2. *for any answer set A of $\pi(D)$, there exists a rational model (σ_0, Φ) of D such that $\sigma_0 = \langle \sigma_{s_0,A}, \Sigma_{s_0,A}\rangle$.*

Proof. Let P_0 be the program consisting of

- the rules about the initial states of $\pi(D)$, i.e., rules (π.12)-(π.16);
- the domain dependent rules, denoted by DD; and
- the language dependent rules, denoted by LD.

It is easy to see that the set of literals $lit(P_0)$ is a splitting set of $\pi(D)$ and the bottom program of $\pi(D)$ w.r.t. $lit(P_0)$ is P_0. From the splitting theorem, we know that

(*) P_0 is consistent (because of the consistency of $\pi(D)$); and

(**) A_0 is an answer set of P_0 iff there exists a solution (A_0, B) of $\pi(D)$ w.r.t. $lit(P_0)$.

From the fact that if (A_0, B) is a solution of $\pi(D)$ w.r.t. $lit(P_0)$ then $A_0 \cap B = \emptyset$, by Observation 2.2 we can conclude that

(***) $\sigma_{s_0,A_0} = \sigma_{s_0,A_0 \cup B}$ and $\Sigma_{s_0,A_0} = \Sigma_{s_0,A_0 \cup B}$.

Therefore, to prove the lemma, it is enough to show that P_0 has an answer set satisfying the conditions of the lemma. We begin with the first item of the lemma.

1. Let (σ_0, Φ) be a rational model of D. We define,

   $$H_1 = \{holds(f, s_0) \mid f \in \sigma_0.1\} \quad \text{and} \quad H_2 = \{\neg holds(f, s_0) \mid f \not\in \sigma_0.1\}.$$

   It is easy to check that the set of literals $X = H_1 \cup H_2 \cup LD \cup DD$ is a splitting set of P_0 and X is the answer set of the bottom program $b_X(P_0)$. By construction of X, we have that

   $$\sigma_0.1 = \sigma_{s_0,X}. \tag{a1}$$

   The partial evaluation of P_0 w.r.t X and $lit(P_0)$, denoted by Q, consists of the following rules:

$$\neg accessible(s, s_0) \quad \leftarrow \quad \text{if } Holds(\bar{f}, s) \in X \quad \text{and } state(s) \in X \quad \text{and } initially(f) \in X \tag{A.22}$$

$$accessible(s, s_0) \quad \leftarrow \quad not\ \neg accessible(s, s_0). \quad \text{if } state(s) \in X \tag{A.23}$$

It is easy to see that the set

$$\begin{array}{rl} Y = & \{\neg accessible(s, s_0) \mid \neg accessible(s, s_0) \text{ is a rule (A.22) of } Q\} \\ & \{accessible(s, s_0) \mid state(s) \in LD, \neg accessible(s, s_0) \text{ does not occur in a rule (A.22) of } Q\} \end{array}$$

is a consistent answer set of Q. We have that $Z_1 = X \cup Y$ is an answer set of P_0, $Y \cap X = \emptyset$, and there exists no literal of the form $Holds(f, s_0) \in Y$. Hence, by Observation 2.2 and (a1) we have that

$\sigma_{s_0,X} = \sigma_{s_0,Z_1}$. (a2)

Furthermore, by the splitting set theorem, we can conclude that there exists a solution w.r.t. $lit(P_0)$, (Z_1, Z_2), of $\pi(D)$ and $A = Z_1 \cup Z_2$ is an answer set of $\pi(D)$. Also, by the splitting theorem, we have that $Z_1 \cap Z_2 = \emptyset$. Therefore, by (***), to prove the lemma, it suffices to show that $\langle \sigma_{s_0,Z_1}, \Sigma_{s_0,Z_1} \rangle = \sigma_0$. From (a2) and (a1), we have that

$\sigma_{s_0,Z_1} = \sigma_0.1$ (a3)

We now show that $\Sigma_{s_0,Z_1} = \sigma_0.2$. We consider two cases:

(1.1) $\sigma_{s,A} \in \Sigma_{s_0,A}$. We need to prove that $\sigma_{s,A}$ is an initial state of D. By our assumption, we have that $accessible(s, s_0) \in Z_1$. By Lemma 2.14 and from the rules (A.22) and (A.23) we have that

$state(s) \in LD$, and
for every f, if $initially(f) \in Z_1$ (resp. $initially(\neg f) \in Z_1$) then $holds(f, s) \in Z_1$ (resp. $\neg holds(f, s) \in Z_1$). This implies that σ_{s,Z_1} is an initial k-state. (a4)

(1.2) s is an initial state. We need to prove that there exists an s' such that $\sigma_{s',Z_1} = s$ and $\sigma_{s',Z_1} \in \Sigma_{s_0,Z_1}$. Since we list all possible states of D in LD and $LD \subseteq Z_1$, there exists an s^* such that $state(s^*) \in Z_1$ and $\sigma_{s^*,Z_1} = s$. Furthermore, because s^* is an initial state, for every f, if $initially(f) \in Z_1$ (resp. $initially(\neg f) \in Z_1$) then $holds(f, s^*) \in Z_1$ (resp. $\neg holds(f^*, s) \in Z_1$). Thus, Q does not contain a rule (A.22) whose head is $\neg accessible(s^*, s_0)$. Therefore, there exists a rule (A.23) in Q whose head is $accessible(s^*, s_0)$ which is applicable in Y. By Lemma 2.14, we conclude that $accessible(s^*, s_0) \in Y \subseteq Z_1$. This proves that $s \in \Sigma_{s_0,Z_1}$. (a5)

It follows from (a4) and (a5) that $\sigma_0.2 = \Sigma_{s_0,Z_1}$. (a6)

The conclusion of the Lemma for the first Item follows from (a6) and (a3).

2. To prove the lemma it suffices to show that $\langle \sigma_{s_0,A}, \Sigma_{s_0,A} \rangle$ is a complete initial k-state of D. By definition, we have that

   $\sigma_{s_0,A} = \{f \mid holds(f, s_0) \in A\}$.

   From the rules (π.12) and (π.13), we have that

   if **initially** $f \in D$ then $holds(f, s_0) \in A$, and (b1)
   if **initially** $\neg f \in D$ then $\neg holds(f, s_0) \in A$. (b2)

   Hence

   if **initially** $f \in D$ then $f \in \sigma_{s_0,A}$, and (b3)
   if **initially** $\neg f \in D$ then $f \notin \sigma_{s_0,A}$. (b4)

   (b3) and (b4) implies that $\sigma_{s_0,A}$ is an initial k-state. (b5)

   We need to prove that $\Sigma_{s_0,A}$ is the set of all initial k-states. We consider two cases:

   (2.1) $\sigma_{s,A} \in \Sigma_{s_0,A}$. We need to prove that $\sigma_{s,A}$ is an initial state of D. By our assumption, we have that $accessible(s, s_0) \in A$. By Lemma 2.14 and the rules (π.15) and (π.16) we have that

   $state(s) \in A$, and
   for every f, if $initially(f) \in A$ (resp. $initially(\neg f) \in A$) then $holds(f, s) \in A$ (resp. $\neg holds(f, s) \in A$). This implies that $\sigma_{s,A}$ is an initial k-state. (b6)

   (2.2) s is an initial state. We need to prove that there exists an s' such that $\sigma_{s',A} = s$ and $\sigma_{s',A} \in \Sigma_{s_0,A}$. Since we list all possible states of D in LD and $LD \subseteq A$, there exists an s^* such that $state(s^*) \in A$ and $\sigma_{s^*,A} = s$. Furthermore, because s is an initial state, for every f, if $initially(f) \in A$ (resp. $initially(\neg f) \in A$) then $holds(f, s^*) \in A$ (resp. $\neg holds(f^*, s) \in A$). Thus, Q does not contain a rule (A.22) whose head is $\neg accessible(s^*, s_0)$. Therefore, there exists a rule (A.23) whose head is $accessible(s^*, s_0)$ which is applicable in A. By Lemma 2.14, we conclude that $accessible(s^*, s_0) \in A$. This proves that $s \in \Sigma_{s_0,A}$. (b7)

   It follows from the two cases (2.1) and (2.2) that $\Sigma_{s_0,A}$ is the set of all initial states of D. (b8)

   The conclusion of the lemma for this case follows from (b5) and (b8).

□

Lemma 2.19 *Let D be a consistent domain description and A be an answer set of $\pi(D)$. Then, for any fluent f, state s, and sequence of actions α*

1. $holds(f, res(\alpha, s)) \in A$ *iff* $f \in Res(\alpha, \sigma_{s,A})$*; and*
2. $\neg holds(f, res(\alpha, s)) \in A$ *iff* $f \notin Res(\alpha, \sigma_{s,A})$.

Proof. We prove the lemma by induction over the length of α. We first prove 1.

(i) $|\alpha| = 0$. By construction of $\sigma_{s,A}$, we have that $holds(f,s) \in A$ iff $f \in \sigma_{s,A}$. The base case is proved.

(ii) Assume that we have proved the lemma for $|\alpha| = n$. We need to show it for $|\alpha| = n+1$. Assume that $\alpha = a_1; \ldots; a_{n+1}$. Let $\beta = a_1; \ldots; a_n$.

Since $\pi(D)$ is consistent (Proposition 9), by Lemma 2.15, we can conclude that A is consistent. Hence, by Lemma 2.14, $holds(f, res(\alpha, s)) \in A$ iff there exists a rule whose head is $holds(f, res(\alpha, s)) \in A$ which is applicable in A. From the construction of $\pi(D)$ we conclude that one of the following cases is true.

(ii1) $holds(f, res(\alpha, s))$ is the head of a rule of the form $(\pi.7)$ which is applicable in A, i.e.,

there exists an ef-proposition a_{n+1} **causes** f **if** $p_1, \ldots, p_m$ in D, and (a1)
$Holds(p_i, res(\beta, s)) \in A$ for $i = 1, \ldots, m$. (a2)

From (a2), by induction hypothesis, we have that p_i holds in $Res(\beta, \sigma_{s,A})$ for $i = 1, \ldots, m$, which, together with the (a1) implies that $f \in Res(a_{n+1}, Res(\beta, \sigma_{s,A}))$, i.e., $f \in Res(\alpha, \sigma_{s,A})$. This proves the inductive step for case (ii1).

(ii2) $holds(f, res(\alpha, s))$ is the head of a rule of the form $(\pi.4)$ which is applicable in A, i.e.,

$holds(f, res(\beta, s)) \in A$, and (b1)
$ab(f, a_{n+1}, res(\beta, s)) \notin A$. (b2)

From (b1) and the inductive hypothesis, we have that $f \in Res(\beta, \sigma_{s,A})$. (b3)

The condition (b2) implies that there exists no ef-proposition
a_{n+1} **causes** $\neg f$ **if** $p_1, \ldots, p_m$ in D such that $Holds(p_i, res(\beta, s)) \in A$ for $i = 1, \ldots, m$.
By induction hypothesis, we have that there exists no ef-proposition
a_{n+1} **causes** $\neg f$ **if** $p_1, \ldots, p_m$ such that p_i holds in $Res(\beta, \sigma_{s,A})$ for $i = 1, \ldots, m$. This, together with (b3), implies that $f \in Res(\alpha, \sigma_{s,A})$, i.e., the inductive step for case (ii2) is proved.

The lemma is proved as the inductive step is proved.

□

Lemma 2.18 shows that there exists an one-to-one correspondence between answer sets of $\pi(D)$ and rational models of D. An answer set A *corresponds* to a model (σ_0, Φ) and vice versa iff $\langle \sigma_{s_0,A}, \Sigma_{s_0,A} \rangle = \sigma_0$.

Lemma 2.20 *Let D be a consistent domain description, A be an answer set of $\pi(D)$, and (σ_0, Φ) be the rational model of D corresponding to A. Then, for any sequence of actions α,*

$$\hat{\Phi}(\alpha, \sigma_0) = \langle \sigma_{res(\alpha,s_0),A}, \Sigma_{res(\alpha,s_0),A} \rangle.$$

Proof. By Lemma 2.18, we have that $M = (\sigma_0, \Phi)$ is constructed in such a way that $\sigma_0 = \langle \sigma_{s_0,A}, \Sigma_{\alpha,A} \rangle$. We will prove that M satisfies the condition of the lemma by induction over the length of α.

The base case is trivial by the construction of M.

We now prove the inductive step. Assume that $\alpha = a_1, \ldots, a_n, a_{n+1}$. Let $\beta = a_1, \ldots, a_n$.

Since $|\beta| = n$, by inductive hypothesis, we have that

$\hat{\Phi}(\beta, \sigma_0) = \langle \sigma_{res(\beta,s_0),A}, \Sigma_{res(\beta,s_0),A} \rangle$. (a)

We consider two cases:

(i) a_{n+1} is a non-sensing action. So, there is no rules of the forms (π.10) and (π.11) of $\pi(D)$ containing a_{n+1} in $\pi(D)$. Furthermore, for each $s \in \Sigma_{res(\alpha,s_0),A}$, there exists a rule of the form

$accessible(res(a_{n+1}, s), res(a_{n+1}, res(\beta, s_0))) \leftarrow accessible(s, res(\beta, s_0))$ (by rule (π.9) of $\pi(D)$)

in $\pi(D)$. By Lemma 2.14, we have that $accessible(res(a_{n+1}, s), res(\alpha, s_0)) \in A$. This implies that,

$\Sigma_{res(\alpha,s_0),A} = \{s \mid accessible(s, res(\beta, s_0)) \in A\}$ (i1)

Furthermore, by Lemma 2.19, we conclude that

$Res(a_{n+1}, \sigma_{res(\beta,s_0),A}) = \sigma_{res(a_{n+1},res(\beta,s_0)),A} = \sigma_{res(\alpha,s_0),A}$. (i2)

It follows from (i1) and (i2) that

$\hat{\Phi}(\alpha, \sigma_0) = \langle \sigma_{res(\alpha,s_0),A}, \Sigma_{res(\alpha,s_0),A} \rangle$. (i3)

The inductive step for case (i) follows from (i3)

(ii) a_{n+1} is a sensing action. Consider a state s such that

$accessible(s, res(a_{n+1}, res(\beta, s_0))) \in A$. (ii1)

Since $\pi(D)$ is consistent (Proposition 9), by Lemma 2.15, A is consistent. Hence, by Lemma 2.14 and the rules (π.10) and (π.11) of $\pi(D)$, we have that (ii1) is possible iff

$accessible(s, res(\beta, s_0)) \in A$, and (ii2)
there exists no k-proposition a_{n+1} **determines** f in D such that
$\neg agree(f, s, res(\beta, s_0)) \in A$. (ii3)

From (ii2) and the inductive hypothesis, we have that (ii2) is possible iff $\sigma_{s,A} \in \Sigma_{res(\beta,s_0),A}$.

Since a_{n+1} is a sensing action, from the construction of $\pi(D)$, we have that
$holds(f, res(a_{n+1}, s)) \in A$ iff $holds(f, s) \in A$. This implies that $\sigma_{s,A} = \sigma_{res(a_{n+1},s),A}$. (ii4)

From the definition of $agree$ we have that (ii3) holds iff

for every k-proposition a_{n+1} **determines** f
$(holds(f, s) \in A$ and $holds(f, res(\beta, s_0)) \in A)$ or
$(\neg holds(f, s) \in A$ and $\neg holds(f, res(\beta, s_0)) \in A)$. (ii5)

Hence, by Lemma 2.19, (ii5) holds iff

$\sigma_{s,A} \in \Sigma_{res(a_{n+1},res(\beta,s_0)),A}$.

This, together with (ii4), implies that (ii1) is possible iff $\sigma_{res(a_{n+1},s),A} \in \Sigma_{res(\alpha,s_0),A}$. (ii6)

The inductive step for this case follows from (ii4) and (ii6).

The inductive step is proved and so is the lemma. □

Lemma 2.21 *Let D be a consistent domain description and A be an answer set of $\pi(D)$. Then, $accessible(s, res(\alpha, s_0)) \in A$ iff there exists a state constant s' such that $state(s') \in A$, $accessible(s', s_0) \in A$, and $s = res(\alpha, s')$.*

Proof. We prove by induction over length of α. Since $\pi(D)$ is consistent (Proposition 9), by Lemma 2.15, A is a consistent answer set of $\pi(D)$. Because of the rules (π.15) and (π.16) and Lemma 2.14, the lemma is trivial for $|\alpha| = 0$.

Assume that we have proved the lemma for $|\alpha| = n$.

We need to prove it for $|\alpha| = n + 1$. Assume that $\alpha = a_1; \ldots; a_{n+1}$. Let $\beta = a_1; \ldots; a_n$.

Therefore, by Lemma 2.14, from $accessible(s, res(\alpha, s_0)) \in A$, we can conclude that there exists a rule in $\pi(D)$ with the head $accessible(s, res(\alpha, s_0))$ whose body is satisfied in A.

By construction of $\pi(D)$, we have that $accessible(s, res(\alpha, s_0)) \in A$ is the head of a rule of the form (π.9), (π.10), or (π.11). This implies that there exists a state s_1 such that $state(s_1) \in A$, $s = res(a_{n+1}, s_1)$ and $accessible(s_1, res(\beta, s_0)) \in A$.

By induction hypothesis, there exists a state s' such that $accessible(s', s_0) \in A$ and $s_1 = res(\beta, s')$. Hence, $s = res(a_{n+1}, s_1) = res(a_{n+1}, res(\beta, s_0)) = res(\alpha, s_0)$. This proves the inductive step. □

Lemma 2.22 *Let D be a consistent domain description, A be an answer set of $\pi(D)$, and (σ_0, Φ) be the model corresponding to A of D. Then, for any fluent f and sequence of action α,*

(i) $knows(f, res(\alpha, s_0)) \in A$ iff f is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$;

(ii) $knows(\neg f, res(\alpha, s_0)) \in A$ iff f is known to be false in $\hat{\Phi}(\alpha, \sigma_0)$.

Proof. By Lemma 2.18 we have that $M = (\sigma_0, \Phi)$ where $\sigma_0 = \langle \sigma_{s_0,A}, \Sigma_{s_0,A} \rangle$. We need to prove that M satisfies the (i) and (ii).

We first prove (i).

Let us assume that $knows(f, res(\alpha, s_0)) \in A$. (a)

Because of the consistency of A, (a) holds iff there exists a rule of the form (π.21) and (π.22) of $\pi(D)$ which is applicable in A. This implies that

for every state s, if $accessible(s, res(\alpha, s_0)) \in A$, then $holds(f, s) \in A$. (b)

From Lemma 2.21, we can conclude that (b) is true iff

for every state s,
if $accessible(s, s_0)$ and $accessible(res(\alpha, s), res(\alpha, s_0)) \in A$, then $holds(f, res(\alpha, s)) \in A$. (c)

By our definition and Lemma 2.19, (c) holds iff

for every $\sigma_{s,A} \in \Sigma_{s_0,A}$, if $\sigma_{res(\alpha,s),A} \in \Sigma_{res(\alpha,s_0),A}$, then $f \in \sigma_{res(\alpha,s),A}$. (d)

By Lemma 2.20, (d) is equivalent to f is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$, i.e., we conclude (i).

(ii) can be proven similarly. □

Lemma 2.23 *Let D be a consistent domain description, $M = (\sigma_0, \Phi)$ be a rational model of D, and A be the answer set corresponding to M. Then, for any fluent f and sequence of actions α*

(i) f is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$ iff $knows(f, res(\alpha, s_0)) \in A$;

(ii) f is known to be false in $\hat{\Phi}(\alpha, \sigma_0)$ iff $knows(\neg f, res(\alpha, s_0)) \in A$.

Proof. By Lemma 2.20, we have that

$$\hat{\Phi}(\alpha, \sigma_0) = \langle \sigma_{res(\alpha,s_0),A}, \Sigma_{res(\alpha,s_0),A} \rangle. \tag{a}$$

By definition, f is known to be true in $\hat{\Phi}(\alpha, \sigma_0)$ iff

for every $s \in \Sigma_{res(\alpha,s_0),A}$, $f \in s$. (b)

By Lemma 2.20 and 2.19, (b) holds iff

for every s, if $accessible(s, res(\alpha, s_0)) \in A$, then $holds(f, s) \in A$. (c)

By construction of $\pi(D)$, (c) implies that $knows(f, res(\alpha, s_0)) \in A$, which proves (i).

(ii) can be proven similarly. □

The next lemma follows directly from the Lemmas 2.22 and 2.23.

Lemma 2.24 *Let D be a consistent domain description and f be a fluent in D. Then, for any sequence of actions α,*

$$\pi(D) \models knows(f, res(\alpha, s_0)) \quad \textit{iff} \quad D \models^r_{\mathcal{A}_K} f \textbf{ after } \alpha.$$

Proof.

1. Assume that $\pi(D) \models knows(f, res(\alpha, s_0))$. We will prove that $D \models^r_{\mathcal{A}_K} f$ **after** α. Assume the contrary, $D \not\models^r_{\mathcal{A}_K} f$ **after** α. This implies that there exists a rational model (σ_0, Φ) of D such that f is not known to be true in $\hat{\Phi}(\alpha, \sigma_0)$. Let A be the answer set of $\pi(D)$ which corresponds to (σ_0, Φ). Then, by Lemma 2.23, $knows(f, res(\alpha, s_0)) \notin A$. This means that $\pi(D) \not\models knows(f, res(\alpha, s_0))$. This contradicts $\pi(D) \models knows(f, res(\alpha, s_0))$. Hence, our assumption is incorrect, i.e., we have proved that $D \models^r_{\mathcal{A}_K} f$ **after** α. So, we can conclude that

   if $\pi(D) \models knows(f, res(\alpha, s_0))$ then $D \models^r_{\mathcal{A}_K} f$ **after** α. (a)

2. Assume that $D \models^r_{\mathcal{A}_K} f$ **after** α. We will prove that $\pi(D) \models knows(f, res(\alpha, s_0))$. Assume the contrary, $\pi(D) \not\models knows(f, res(\alpha, s_0))$. This implies that there exists an answer set A of $\pi(D)$ such that $knows(f, res(\alpha, s_0)) \notin A$. Let (σ_0, Φ) be the rational model of D which corresponds to A. Then, by Lemma 2.22, f is not known to be true in $\hat{\Phi}(\alpha, \sigma_0)$. This means that $D \not\models^r_{\mathcal{A}_K} f$ **after** α. This contradicts $D \models^r_{\mathcal{A}_K} f$ **after** α. Hence, our assumption is incorrect, i.e., we have proved that $\pi(D) \models knows(f, res(\alpha, s_0))$. So, we can conclude that

   if $D \models^r_{\mathcal{A}_K} f$ **after** α then $\pi(D) \models knows(f, res(\alpha, s_0))$. (b)

The lemma follows from (a) and (b). □

In the next lemma we prove some properties of the predicate *apply* which will be used in proving the Proposition 3.

Lemma 2.25 *Let A be an answer set of $\pi(D)$. Then,*

(i) for any situation term s and conditional plan c, if $apply(c, s, s') \in A$ and $apply(c, s, s'') \in A$ then $s' = s''$;

(ii) for any situation term s and conditional plans c and c', if $apply(c, s, s') \in A$ and $apply(c', s', s'') \in A$ then $apply([c; c'], s, s'') \in A$.

Proof. We prove (i) by induction over the number of case plans in c, $count(c)$.

Base case: $count(c) = 0$. Since $\pi(D)$ is consistent (Proposition 9), by Lemma 2.15, we have that A is consistent. Because c is a sequence of actions and only rules (π.20) and (π.19) are applicable for sequence of actions, we can conclude that if $apply(c, s, s') \in A$ then $s' = res(c, s)$. Similarly, if $apply(c, s, s'') \in A$ then $s'' = res(c, s)$. Hence, $s' = s'' = res(c, s)$. This proves the base case.

Inductive step: Assume that we have proved (i) for $count(c) \leq k$. We need to prove it for $count(c) = k + 1$. Let us assume that $c = c_1; \ldots; c_n$ where c_i is either a sequence of actions or a case plan and if c_i is a sequence of actions then c_{i+1} is a case plan (for $1 \leq i < n - 1$). We consider two cases:

(i1) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

Here, we have two cases:

Case 1: There exists some j, $(1 \leq j \leq l)$, such that $knows\_list(\varphi_j, s) \in A$. Then, rules ($\pi$.29)-($\pi$.31) and the consistency of A imply that $apply(c, s, s') \in A$ if there exists u' such that $apply(p_j, s, u') \in A$ and $apply(c', u', s') \in A$ where $c' = c_2, \ldots, c_n$. Since the φ_j are mutual exclusive, we can prove similar conclusion for s'', i.e., there exists a v' such that $apply(p_j, s, v') \in A$ and $apply(c', v', s'') \in A$. Since $count(p_j) \leq count(c) - 1$ and $count(c') \leq count(c) - 1$, by inductive hypothesis, we can conclude that $u' = v'$, and hence, $s' = s''$. This proves the inductive step for this case.

Case 2: There exists no j, $(1 \leq j \leq l)$, such that $knows\_list(\varphi_j, s) \in A$. Then, rules ($\pi$.29)-($\pi$.31) imply that $apply(c, s, \perp) \in A$ which proves the inductive step for this case because $s' = s'' = \perp$.

(i2) c_1 is a sequence of actions. Then c_2 is a case plan. Since c_1 is a sequence of actions, we have that $apply(c, s, s') \in A$ iff $apply(c', res(c_1, s), s') \in A$ where $c' = c_2, \ldots, c_n$. Similarly, $apply(c, s, s'') \in A$ iff $apply(c', res(c_1, s), s'') \in A$. Since c_1 is a sequence of actions, we have that $count(c) = count(c')$. Since c' begins with a case plan, from the case (i1), we know that $s' = s''$ which proves the inductive step for case (i2).

The inductive step follows from the two cases (i1) and (i2). So, (i) is proved.

The proof of (ii) can be done similar to the proof of (i) and is omitted here. $\square$

The next two lemmas are similar to the Lemmas 2.6-2.7.

Lemma 2.26 *Assume that $c = c_1, \ldots, c_n$ is a conditional plan where $c_1, \ldots, c_n$ is a sequence of conditional plans satisfying the conditions of Observation 3.1. Let c_1 be a case plan of the form*

Case
$\quad \varphi_1 \rightarrow p_1$
$\quad \vdots$
$\quad \varphi_l \rightarrow p_l$
Endcase

and s be a situation term. Assume that A be an answer set of $\pi(D)$ and $knows(\varphi_j, s) \in A$ for some j, $1 \le j \le l$. Then, $apply(c, s, s') \in A$ iff $apply(c^*, s, s') \in A$ where $c^* = p_j; c_2; \ldots; c_n$.

Proof. Let $c' = c_2; \ldots; c_n$. As $\pi(D)$ is consistent (Proposition 9) we have that A is consistent. Then, by rule (π.29) and Lemma 2.14, the assumption that $knows(\varphi_j, s) \in A$ we have that

$apply(c, s, s') \in A$ iff
$\exists s''$ such that $apply(p_j, s, s'') \in A$ and $apply(c', s'', s') \in A$ iff
$apply(c^*, s, s') \in A$ (by Item (ii) of Lemma 2.25) (1)

The lemma follows from (1). □

Lemma 2.27 *Assume that $c = c_1, \ldots, c_n$ is a conditional plan where $c_1, \ldots, c_n$ is a sequence of conditional plans satisfying the conditions of Observation 3.1. Let c_1 be a sequence of actions and c_2 be a case plan of the form*

Case
$\quad \varphi_1 \rightarrow p_1$
$\quad \vdots$
$\quad \varphi_l \rightarrow p_l$
Endcase

and s be a situation term. Let A be an answer set of $\pi(D)$ such that $knows(\varphi_j, res(c_1, s)) \in A$ for some j, $1 \le j \le l$. Then, $apply(c, s, s') \in A$ iff $apply(c^*, s, s') \in A$ where $c^* = c_1; p_j; \ldots; c_n$.

Proof. Let $c' = c_2; \ldots; c_n$. Since $\pi(D)$ is consistent (by Proposition 9), A is consistent. Hence, by Lemma 2.14 and rule (π.20) we have that $apply(c, s, s') \in A$ iff $apply(c', res(c_1, s), s') \in A$. (1)

Let $c'' = p_j; \ldots; c_n$. Then, by Lemma 2.26, we have that
$apply(c', res(c_1, s), s') \in A$ iff $apply(c'', res(c_1, s), s') \in A$. (2)

By rule (π.20), we have that $apply(c'', res(c_1, s), s') \in A$ iff $apply(c^*, s, s') \in A$. (3)

The lemma follows from (1)-(3). □

We now extend the result of Lemmas 2.22 and 2.23 to conditional plans.

Lemma 2.28 *Let D be a consistent domain description. Then,*

(i) Let A be an answer set of $\pi(D)$ and (σ_0, Φ) be the rational model of D which corresponds to A. Then, for any fluent f and conditional plan c,

(i.1) $knows\_after(f, c, s_0) \in A$ iff f is known to be true in $\hat{\Phi}(c, \sigma_0)$;

(i.1) $knows(\neg f, c, s_0) \in A$ iff f is known to be false in $\hat{\Phi}(c, \sigma_0)$.

(ii) *Let* (σ_0, Φ) *be a rational model and* A *be the answer set of* $\pi(D)$ *which corresponds to* (σ_0, Φ). *Then, for any fluent* f *and conditional plan* c,

(ii.1) f *is known to be true in* $\hat{\Phi}(c, \sigma_0)$ *iff* $knows\_after(f, c, s_0) \in A$;

(ii.2) f *is known to be false in* $\hat{\Phi}(c, \sigma_0)$ *iff* $knows(\neg f, c, s_0) \in A$.

Proof. We first prove (i.1). We prove that

(*) $knows\_after(f, c, s_0) \in A$ iff f is known to be true in $\hat{\Phi}(c, \sigma_0)$.

By Observation 3.1, we know that c can be represented as a sequence of conditional plans $c_1, \ldots, c_n$ where c_i is either a sequence of actions or a case plan and for every $i < n$ if c_i is a sequence of actions then c_{i+1} is a case plan. We prove the lemma by induction over $count(c)$, the number of case plans in c. As in previous lemmas, we can c

Base case: $count(c) = 0$. Then, c is a sequence of actions. By Lemma 2.22, we conclude the base case.

Inductive step: Assume that we have proved the lemma for $count(c) \leq k$. We need to prove it for $count(c) = k + 1$. Assume that $c = c_1; \ldots; c_n$. We consider two cases:

(I) $knows\_after(f, c, s_0) \in A$. We will prove that f is known to be true in $\hat{\Phi}(c, \sigma_0)$. Here, we have two cases too:

(Ia) c_1 is a case plan. Assume that c_1 is the following case plan

Case
$\varphi_1 \rightarrow p_1$
$\vdots$
$\varphi_l \rightarrow p_l$
Endcase

We first prove that there exists a j, $1 \leq j \leq l$, such that $knows\_list(\varphi_j, s_0) \in A$.

Assume the contrary, i.e., there exists no j, $1 \leq j \leq l$, such that $knows\_list(\varphi_j, s_0) \in A$. Then, by Lemma 2.14 and from rules (π.29)-(π.31), we have that $apply(c, s_0, \perp) \in A$. Thus, none of the rule of the type (π.25) is applicable in A, and hence, by Lemma 2.14, we have that $knows\_after(f, c, s_0) \not\in A$. This contradicts our assumption that $knows\_after(f, c, s_0) \in A$, i.e., we have proved that there exists j, $1 \leq j \leq l$, such that $knows\_list(\varphi_j, s_0) \in A$. (a1)

It follows from (a1) and Lemma 2.24 that φ_j holds in σ_0. Let $c^* = p_j; c_2; \ldots; c_n$. By definition of $\hat{\Phi}$, we have that

$$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0). \quad \text{(a2)}$$

Since $knows\_list(\varphi_j, s_0) \in A$, by Lemma 2.26, we have that
$apply(c, s_0, s) \in A$ iff $apply(c^*, s_0, s) \in A$. Thus, from rule (π.25), we conclude that

$$knows\_after(f, c, s_0) \in A \text{ iff } knows\_after(f, c^*, s_0) \in A. \quad \text{(a3)}$$

Since $count(c) \geq count(c^*) + 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis, we conclude that,

f is known to be true in $\hat{\Phi}(c^*, \sigma_0)$ iff $knows\_after(f, c^*, s_0) \in A$. (a4)

From (a2)-(a4) we conclude that

f is known to be true in $\hat{\Phi}(c, \sigma_0)$. (a5)

The conclusion for case (Ia) follows from (a5).

(Ib) c_1 is a sequence of action. Then, c_2 is a case plan. Assume that c_2 is the case plan

Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
Endcase

Similar to the previous case, we can prove that there exists a j, $1 \leq j \leq l$, such that $knows\_list(\varphi_j, res(c_1, s_0)) \in A$. Let $c^* = c_1; p_j; c_3; \ldots; c_n$. Then, by Lemma 2.27, we have hat

$apply(c, s_0, s) \in A$ iff $apply(c^*, s_0, s) \in A$. (b1)

Furthermore, since c_1 is a sequence of action and $knows\_list(\varphi_j, res(c_1, s_0)) \in A$, by Lemma 2.24, we conclude that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. Hence,

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0)$. (b2)

Since $count(c) \geq count(c^*) + 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis, we conclude that,

f is known to be true in $\hat{\Phi}(c^*, \sigma_0)$ iff $knows\_after(f, c^*, s_0) \in A$. (b3)

From (b1)-(b3) we conclude that

f is known to be true in $\hat{\Phi}(c, \sigma_0)$. (b4)

The conclusion for case (Ib) follows from (b4).

It follows from the two cases (Ia) and (Ib) that if $knows\_after(f, c, s_0) \in A$ then f is known to be true in $\hat{\Phi}(c, \sigma_0)$ for $count(c) = k + 1$.

(II) f is known to be true in $\hat{\Phi}(c, \sigma_0)$. We will prove that $knows\_after(f, c, s_0) \in A$. Again, we have two cases:

(IIa) c_1 is a case plan. Assume that c_1 is the following case plan

Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
Endcase

Since f is known to be true in $\hat{\Phi}(c, \sigma_0)$, we have that that there exists a j, $1 \leq j \leq l$, such that φ_j holds in σ_0. (c1)

Let $c^* = p_j; c_2; \ldots; c_n$. We have that

$$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0). \tag{c2}$$

It follows from (c1) and Lemma 2.24 that $knows\_list(\varphi_j, s_0) \in A$. By Lemma 2.26, we have that $apply(c, s_0, s) \in A$ iff $apply(c^*, s_0, s) \in A$. Thus, from rule (π.25), we conclude that

$$knows\_after(f, c, s_0) \in A \text{ iff } knows\_after(f, c^*, s_0) \in A. \tag{c3}$$

Since $count(c) \geq count(c^*) + 1$, we have that $count(c^*) \leq k$. Therefore, by inductive hypothesis, we conclude that,

$$f \text{ is known to be true in } \hat{\Phi}(c^*, \sigma_0) \text{ iff } knows\_after(f, c^*, s_0) \in A. \tag{c4}$$

From (c2)-(c4) we conclude that

$$knows\_after(f, c, s_0) \in A. \tag{c5}$$

The conclusion for case (IIa) follows from (c5).

(IIb) c_1 is a sequence of action. Then, c_2 is a case plan. Assume that c_2 is the case plan

Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
Endcase

Similar to the previous case, we can prove that there exists a j, $1 \leq j \leq l$, such that φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$. Let $c^* = c_1; p_j; c_3; \ldots; c_n$. By definition of $\hat{\Phi}$, we have that

$$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \sigma_0). \tag{d1}$$

Furthermore, from φ_j holds in $\hat{\Phi}(c_1, \sigma_0)$ and Lemma 2.24, we conclude that $knows\_list(\varphi_j, res(c_1, s_0)) \in A$. Hence, by Lemma 2.27, we have hat

$$apply(c, s_0, s) \in A \text{ iff } apply(c^*, s_0, s) \in A. \tag{d2}$$

Since $count(c) \geq count(c^*) + 1$, we have that $count(c^*) \leq k$. Thus, by inductive hypothesis, we conclude that,

$$f \text{ is known to be true in } \hat{\Phi}(c^*, \sigma_0) \text{ iff } knows\_after(f, c^*, s_0) \in A. \tag{d3}$$

From (d1)-(d3) we conclude that

$$knows\_after(f, c, s_0) \in A. \tag{d4}$$

The conclusion for case (IIb) follows from (d4).

It follows from the two cases (IIa) and (IIb) that if $knows\_after(f, c, s_0) \in A$ then f is known to be true in $\hat{\Phi}(c, \sigma_0)$ for $count(c) = k + 1$.

The inductive step follows from the two case (I) and (II). So, we have proved (i.1).

Similarly, we can prove (i.2), (ii.1), and (ii.2). The lemma is proved. □

We now prove the Proposition 3.

Proposition 3 (Soundness and Completeness of $\pi(D)$) Let D be a consistent domain description and f be a fluent in D. Then, for any conditional plan c,

$$\pi(D) \models knows\_after(f, c, s_0) \quad \text{iff} \quad D \models^r_{\mathcal{A}_K} f \ \mathbf{after} \ \ c.$$

Proof.

1. Assume that $\pi(D) \models knows\_after(f, c, s_0)$. We will prove that $D \models^r_{\mathcal{A}_K} f$ **after** c. Assume the contrary, $D \not\models^r_{\mathcal{A}_K} f$ **after** c. This implies that there exists a rational model (σ_0, Φ) such that f is not known to be true in $\hat{\Phi}(\alpha, \sigma_0)$. Let A be the corresponding answer set of (σ_0, Φ). Then, by Lemma 2.28 (Item (i.1)), $knows\_after(f, c, s_0) \notin A$. This means that $\pi(D) \not\models knows\_after(f, c, s_0)$. This contradicts $\pi(D) \models knows\_after(f, c, s_0)$. Hence, our assumption is incorrect, i.e., we have proved that $D \models^r_{\mathcal{A}_K} f$ **after** c. So, we can conclude that

   if $\pi(D) \models knows\_after(f, c, s_0)$ then $D \models^r_{\mathcal{A}_K} f$ **after** c. (a)

2. Assume that $D \models^r_{\mathcal{A}_K} f$ **after** c. We will prove that $\pi(D) \models knows\_after(f, c, s_0)$. Assume the contrary, $\pi(D) \not\models knows\_after(f, c, s_0)$. This implies that there exists an answer set A of $\pi(D)$ such that $knows\_after(f, c, s_0) \notin A$. Let (σ_0, Φ) be the corresponding rational model of D. Then, by Lemma 2.22, f is not known to be true in $\hat{\Phi}(c, \sigma_0)$. This means that $D \not\models^r_{\mathcal{A}_K} f$ **after** c. This contradicts $D \models^r_{\mathcal{A}_K} f$ **after** c. Hence, our assumption is incorrect, i.e., we have proved that $\pi(D) \models knows\_after(f, c, s_0)$. So, we can conclude that

   if $D \models^r_{\mathcal{A}_K} f$ **after** c then $\pi(D) \models knows\_after(f, c, s_0)$. (b)

The proposition follows from (a) and (b). □

Appendix D

In this appendix we will prove the Proposition 8. As our approach is very similar to the approach of Lobo et al., to avoid confusion, a model of the domain D in our approach is referred to simply as a model while a model of D in the approach of Lobo et al. is called a LTM-model. For a sequence of action $\alpha = a_1, \ldots, a_n$, we call the sequences $a_1, \ldots, a_i$ (for $i = 1, \ldots, n$) *prefixes* of α.

Lemma 4.29 *Let D be a consistent domain, Σ_0 be the set of initial states of D, and Φ be a LTM-model of D. Then, for any pair of sequence of actions α and state $s \in \Gamma_\Phi(\alpha, \Sigma_0)$, there exists a rational model (σ_0, Ψ) of D such that*

(i) $\hat{\Psi}(\alpha, \sigma_0).1 = s$ *and* $\Gamma_\Phi(\alpha, \Sigma_0) = \hat{\Psi}(\alpha, \sigma_0).2$;

(ii) for every prefix β of α, $\Gamma_\Phi(\beta, \Sigma_0) = \hat{\Psi}(\beta, \sigma_0).2$.

Proof. We prove the lemma by induction over the length of α.

Base case: $|\alpha| = 0$. We have that

$\Gamma_\Phi([], \Sigma_0) = \Sigma_0$ (by Definition of Γ_Φ) (a)

Let s be an arbitrary state belonging to Σ_0, by Definition 3, we have that $\langle s, \Sigma_0 \rangle$ is an initial k-state of D. Furthermore, by the assumption of the lemma, Σ_0 is the set of all initial states of D.

Hence, $(\langle s, \Sigma_0\rangle, \Psi)$ is a rational model of D where Ψ is defined as in Definition 2. Furthermore, by Definition of the extended function of D, we have that

$\hat{\Psi}([], \langle s, \Sigma_0\rangle) = \langle s, \Sigma_0\rangle$. (b)

From (a) and (b), we have that $(\langle s, \Sigma_0\rangle, \Psi)$ is a rational model of D that satisfies the lemma for $\alpha = []$, i.e., the base case is proved.

Inductive step: Assume that we have proved the lemma for $|\alpha| = n$. We need to prove for $|\alpha| = n+1$. Let $\alpha = \beta; a$ where $|\beta| = n$.

Let $\Gamma_\Phi(\beta, \Sigma_0) = \Sigma$.

Consider a state s in Σ. (c)

We consider two cases:

(i) a is a non-sensing action. Then, by definition of Γ_Φ, there exists a state

$u \in \Sigma$ (d1)
such that $s = Res(a, u)$. (d2)

Because $|\beta| = n$, from the inductive hypothesis, we know that there exists a rational model (σ_0, Ψ) satisfying that

$\hat{\Psi}(\beta, \sigma_0).1 = u$, (d3)
and for every prefix γ of β, $\Gamma_\Phi(\gamma, \Sigma_0) = \hat{\Psi}(\gamma, \sigma_0).2$. (d4)

We have that
$\hat{\Psi}(\alpha, \sigma_0) = \Psi(a, \hat{\Psi}(\beta, \sigma_0))$ (by Definition)
$\Rightarrow \hat{\Psi}(\alpha, \sigma_0) = \langle Res(a, u), \{Res(a, s') \mid s' \in \Gamma_\Phi(\beta, \Sigma_0)\}\rangle$ (by (d4) and Definition)
$\Rightarrow \hat{\Psi}(\alpha, \sigma_0).1 = s$, and (by Definition of Ψ and (d2)) (d5)
$\Rightarrow \hat{\Psi}(\alpha, \sigma_0).2 = \Gamma_\Phi(\alpha, \Sigma_0)$. (d5)

From (d3), (d5), and (d6), we conclude the inductive step for case (i).

(ii) a is a sensing action. Assume that a **determines** $f_1, \ldots, a$ **determines** f_m be the k-propositions containing a in D. By definition of Γ_Φ we have that
$\Gamma_\Phi(\alpha, \Sigma_0) = \Phi(a, \Gamma_\Phi(\beta, \Sigma_0)) = \Phi(a, \Sigma)$.

Let us assume that $\Phi(a, \Sigma) = \bigcap_{i=1}^{m} \Sigma_i = \Sigma'$ where Σ_i is a situation which is f_i-compatible with Σ. It is easy to see that

(e1) for any i ($i = 1, \ldots, m$) and any pair of states s and s' in Σ_i, s agrees with s' on f_i.

From (e1) we can easily prove the following:

(e2) for any pair of states s and s' in Σ', s agrees with s' on each f_i (for $i = 1, \ldots, m$);

(e3) if a state $s \in \Sigma$ agrees with a state $s' \in \Sigma'$ on each f_i (for $i = 1, \ldots, m$), then $s \in \Sigma'$.

Now, let us consider a state $s \in \Sigma'$. Then, by definition, we have that $s \in \Sigma$. Thus, by inductive hypothesis, we know that there exists a rational model (σ_0, Ψ) satisfying that

$\hat{\Psi}(\beta, \sigma_0).1 = s,$ (e4)
and for every prefix γ of β, $\Gamma_\Phi(\gamma, \Sigma_0) = \hat{\Psi}(\gamma, \sigma_0).2$. (e5)

We then have that

$\hat{\Psi}(\beta, \sigma_0).1 = \hat{\Psi}(\alpha, \sigma_0).1 = s$, and (by Definition of Ψ and (e4)) (e6)
$\hat{\Psi}(\alpha, \sigma_0).2 = \{s' \mid s' \in \hat{\Psi}(\beta, \sigma_0).2,\ s$ agrees with s on each f_i for $i = 1, \ldots, m\}$ (by Definition)
$\Rightarrow \hat{\Psi}(\alpha, \sigma_0).2 = \{s' \mid s' \in \Sigma,\ s$ agrees with s on each f_i for $i = 1, \ldots, m\}$ (by (e5))
$\Rightarrow \hat{\Psi}(\alpha, \sigma_0).2 = \Sigma'$. (by (e3)) (e7)

From (e6), (e7), and (e5), we conclude the inductive step for case (ii).

As the inductive step is proved, the lemma is proved. □

Lemma 4.30 *Let D be a consistent domain description, Σ_0 be the set of its initial states, and $(\langle s_0, \Sigma_0\rangle, \Psi)$ be a rational model of D. Then, for any sequence of actions α there exists a LTM-model Φ of D such that $\Gamma_\Phi(\gamma, \Sigma_0) = \hat{\Psi}(\gamma, \sigma_0).2$ for every prefix γ of α.*

Proof. We prove the lemma by induction over $|\alpha|$.

Base case: We have that

$\hat{\Psi}([], \langle s_0, \Sigma_0\rangle) = \langle s_0, \Sigma_0\rangle$. (a)

and for any LTM-model Φ,

$\Gamma_\Phi([], \Sigma_0) = \Sigma_0$. (b)

The base case of the lemma follows from (a) and (b)

Inductive step: Assume that we have proved the lemma for $|\alpha| = n$.

We need to prove it for $|\alpha| = n + 1$.

Assume that $\alpha = a_1, \ldots, a_n, a_{n+1}$. Let $\alpha_0 = []$ and $\alpha_i = a_1, \ldots, a_i$ for $i = 1, \ldots, n$.

By inductive hypothesis, there exists a LTM-model Φ of D such that

$\Gamma_\Phi(\gamma, \Sigma_0) = \hat{\Psi}(\gamma, \sigma_0).2$, for every prefix γ of α_n. (By Definition of Γ_Φ) (c)

We consider two cases:

(i) a_{n+1} is a non-sensing action. By definitions, we have that

$\Gamma_\Phi(\alpha, \Sigma_0) = \{Res(a_{n+1}, s') \mid s' \in \Gamma_\Phi(\alpha_n, \Sigma_0)\}$
$\Rightarrow \Gamma_\Phi(\alpha, \Sigma_0) = \{Res(a_{n+1}, s') \mid s' \in \hat{\Psi}(\alpha_n, \sigma_0).2\}$ (by (c), for $\gamma = \alpha_n$)
$\Gamma_\Phi(\alpha, \Sigma_0) = \hat{\Psi}(\alpha, \sigma_0).2$ (by Definition of $\hat{\Psi}$) (d)

The inductive step for this case follows from (d) and (c).

(ii) a_{n+1} is a sensing action. Let a **determines** $f_1, \ldots, a$ **determines** f_m be the k-propositions containing a in D. We consider two cases:

1. there exists i, $1 \leq i \leq n$ such that $a_i = a_{n+1}$ and $\hat{\Psi}(\alpha_{i-1}, \sigma_0).2 = \Sigma$. (e1)

   From the definition of the transition function of D, we know that the number of states in $\hat{\Psi}(\alpha_n, \sigma_0).2$ will be less than the number of states of $\hat{\Psi}(\alpha_{i-1}, \sigma_0).2$ if there exists a state $s \in \hat{\Psi}(\alpha_{i-1}, \sigma_0).2$ which does not agree with $\hat{\Psi}(\alpha_{i-1}, \sigma_0).1$ on $\{f_1, \ldots, f_n\}$. Therefore, from (e1), we can conclude that every state $s \in \hat{\Psi}(\alpha_{i-1}, \sigma_0).2$ agrees with $\hat{\Psi}(\alpha_{i-1}, \sigma_0).1$ on $\{f_1, \ldots, f_n\}$.

   This implies that $\hat{\Psi}(\alpha_i, \sigma_0).2 = \hat{\Psi}(\alpha_{i-1}, \sigma_0).2 = \Sigma$. (e2)

   Thus, by (c), we have that $\Gamma_\Phi(\alpha_i, \Sigma_0) = \Sigma$. This implies that $\Phi(a_i, \Sigma) = \Sigma$. Hence, we have that $\Gamma_\Phi(\alpha, \Sigma_0) = \Phi(a_{n+1}, \Sigma) = \Sigma$. Thus, Φ itself is a LTM-model satisfying the conclusion of the inductive step for this case.

2. for every $i = 1, \ldots, n$, $a_i \neq a$ or $(a_i = a_{n+1} \wedge \Gamma_\Phi(\alpha_{i-1}, \Sigma_0) \neq \Sigma)$. This means that $\Phi(a_{n+1}, \Sigma)$ is not used in computing $\Gamma_\Phi(\alpha_i, \Sigma_0)$ for $i = 1, \ldots, n$. Then, we will define a new LTM-model Φ' satisfying the conclusion of the lemma. First, we define a sequence of situations $\Sigma_1, \ldots, \Sigma_m$ by

   $\Sigma_i = \hat{\Psi}(\alpha, \sigma_0).2 \cup \{s \mid s \in \hat{\Psi}(\beta, \sigma_0).2 \text{ such that } f_i \in s \text{ iff } f_i \text{ is true in } \hat{\Psi}(\beta, \sigma_0).2\}$.

   It is easy to see that Σ_i is a f_i-compatible situation with $\Gamma_\Phi(\beta, \Sigma_0)$ and $\cap_{i=1}^m \Sigma_i = \hat{\Psi}(\alpha, \sigma_0).2$.

   We then define the model Φ' as follows.

   (e3) $\Phi'(b, \Sigma') = \Phi(b, \Sigma')$ for every action $b \neq a_{n+1}$ or $(b = a_{n+1}) \wedge \Sigma' \neq \Sigma$, and

   (e4) $\Phi'(a_{n+1}, \Sigma) = \cap_{i=1}^m \Sigma_i$.

   It is easy to verify that for every prefix α_i of α_n, $\Gamma_{\Phi'}(\alpha_i, \Sigma_0) = \Gamma_\Phi(\alpha_i, \Sigma_0)$, and $\Gamma_{\Phi'}(\alpha, \Sigma_0) = \Sigma$. This proves the inductive step for this case.

As the inductive step is proved, the lemma is proved. □

We now extend the result of the Lemmas 4.29 and 4.30 to conditional plans. We will need the following lemma which follows immediately from Lemmas 4.29 and 4.30.

Lemma 4.31 *Let D be a consistent domain description and c be a conditional plan. Then,*

1. *for every rational model (σ_0, Ψ) of D, if $\hat{\Psi}(c, \sigma_0) = \perp$, there exists a LTM-model Φ of D such that $\Gamma_\Phi(c, \Sigma_0) = \perp$;*

2. *for every LTM-model Φ of D, if $\Gamma_\Phi(c, \Sigma_0) = \perp$, there exists a rational model (σ_0, Ψ) of D such that $\hat{\Psi}(c, \sigma_0) = \perp$.*

Lemma 4.32 Let D be a consistent domain description, (σ_0, Ψ) be a rational model of D, and $c = c_1; \ldots; c_n$ be a conditional plan such that $\perp \neq \hat{\Phi}(c, \sigma_0)$. Then, there exists a LTM-model Φ of D such that $\hat{\Psi}([c_1; \ldots; c_t], \sigma_0).2 = \Gamma_\Phi([c_1; \ldots; c_t], \Sigma_0)$ for $t = 1, \ldots, n$.

Proof. We prove the lemma by induction over $case(c)$, the number of case plans in c.

The base case, $case(c) = 0$, is proved by Lemma 4.29.

We now prove the inductive step. Assume that the lemma is shown for $case(c) \leq k$, we prove the lemma for $case(c) = k + 1$.

We consider two cases:

- c_1 is a case plan. Assume that c_1 is the following plan

  Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
  Endcase

  From the assumption that $\perp \neq \hat{\Psi}(c, \sigma_0)$, we can conclude that there exists j, $1 \leq j \leq l$, such that φ_j holds in σ_0. Let $c^* = p_j; c_2; \ldots; c_i$. Then, by definition of $\hat{\Psi}$ we have that

  $$\hat{\Psi}(c^*, \sigma) = \hat{\Psi}(c, \sigma). \tag{a1}$$

  Since $case(c) \geq case(c^*) + 1$, we have that $case(c^*) \leq k$. It follows from the induction hypothesis that there exists a LTM-model Φ such that

  $$\hat{\Psi}(c^*, \sigma_0).2 = \Gamma_\Phi(c^*, \Sigma_0), \tag{a2}$$
  $$\hat{\Psi}(p_j, \sigma_0).2 = \Gamma_\Phi(p_j, \Sigma_0), \text{ and} \tag{a3}$$
  $$\hat{\Psi}([p_j; c_2; \ldots; c_t], \sigma_0).2 = \Gamma_\Phi([p_j; c_2; \ldots; c_t], \Sigma_0) \text{ for } t = 2, \ldots, n-1. \tag{a4}$$

  Since φ_j holds in Σ_0, we have that

  $$\Gamma_\Phi(c, \Sigma_0) = \Gamma_\Phi(c^*, \Sigma_0). \tag{a5}$$

  From (a5), (a2), and (a1) we have that $\Gamma_\Phi(c, \Sigma_0) = \hat{\Psi}(c, \sigma_0).2$. (a6)

  Furthermore, because of φ_j holds in Σ_0, we have that

  $$\Gamma_\Phi(c_1, \Sigma_0) = \Gamma_\Phi(p_j, \Sigma_0), \text{ and}$$
  $$\Gamma_\Phi([c_1; c_2; \ldots; c_t], \Sigma_0) = \Gamma_\Phi([p_j; c_2; \ldots; c_t], \Sigma_0) \text{ for } t = 2, \ldots, n-1. \tag{a7}$$

  and

  $$\hat{\Psi}(c_1, \sigma_0).2 = \hat{\Psi}(p_j, \sigma_0).2, \text{ and}$$
  $$\hat{\Psi}([c_1; c_2; \ldots; c_t], \sigma_0).2 = \hat{\Psi}([p_j; c_2; \ldots; c_t], \sigma_0).2 \text{ for } t = 2, \ldots, n-1. \tag{a7}$$

  From (a3), (a6), and (a7), we have that $\Gamma_\Phi([c_1; c_2; \ldots; c_t], \Sigma_0) = \hat{\Psi}([c_1; c_2; \ldots; c_t], \sigma_0).2$ for $t = 1, \ldots, n-1$. (a8)

  The conclusion of the inductive step for this case follows from (a7) and (a8).

- c_1 is a sequence of actions. Then, c_2 is a case plan. Assume that c_2 is the following plan

  Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
  Endcase

  Let $c^* = c_3; \ldots; c_n$. By definition we have that $\hat{\Phi}(c, \sigma_0) = \hat{\Phi}(c^*, \hat{\Phi}(c_2, \hat{\Phi}(c_1, \sigma_0)))$.

Since $\perp \neq \hat{\Phi}(c, \sigma_0)$, we can conclude that there exists a j, $1 \leq j \leq l$, such that φ_j holds in $\hat{\Phi}(c_1, \sigma_0))$. This implies that

$\hat{\Phi}(c, \sigma_0) = \hat{\Phi}([c_1; p_j; c^*], \sigma_0)$.

Since $c' = c_1; p_j; c^*$ is a conditional plan with $case(c') \leq k$, by induction hypothesis, there exists a rational model satisfying the conditions (i) and (ii) of the Lemma w.r.t. c'. Similar to the first case, we can easily verify that Φ satisfies (i) and (ii) w.r.t. c. The inductive step is proved.

$\square$

Lemma 4.33 Let D be a consistent domain description, Φ be a LTM-model of D, and $c = c_1; \ldots; c_n$ be a conditional plan such that $\perp \neq \Gamma_\Phi(c, \Sigma_0)$. Then, there exists a rational model (σ_0, Ψ) of D such that $\hat{\Psi}([c_1; \ldots; c_j], \sigma_0).2 = \Gamma_\Phi([c_1; \ldots; c_j], \sigma_0)$ for $j = 1, \ldots, n$.

Proof. We prove the lemma by induction over $case(c)$, the number of case plans in c.

The base case, $case(c) = 0$, is proved by Lemma 4.30.

We now prove the inductive step. Assume that the lemma is shown for $case(c) \leq k$, we prove the lemma for $case(c) = k + 1$.

We consider two cases:

- c_1 is a case plan. Assume that c_1 is the following plan

  Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
  Endcase

  From the assumption that $\perp \neq \Gamma_\Phi(c, \Sigma_0)$, we can conclude that there exists j, $1 \leq j \leq l$, such that φ_j holds in Σ_0. Let $c^* = p_j; c_2; \ldots; c_n$. Then, by definition of Γ_Φ we have that

  $\Gamma_\Phi(c, \Sigma_0) = \Gamma_\Phi(c^*, \Sigma_0)$. (a1)

  Since $case(c) \geq case(c^*) + 1$, we have that $case(c^*) \leq k$. It follows from the induction hypothesis that there exists a rational model (σ_0, Φ) of D such that

  $\hat{\Psi}(p_j, \sigma_0).2 = \Gamma_\Phi(p_j, \Sigma_0)$, and (a2)
  $\hat{\Psi}([p_j; c_2; \ldots; c_t], \sigma_0).2 = \Gamma_\Phi([p_j; c_2; \ldots; c_t], \Sigma_0)$ for $t = 2, \ldots, n$. (a3)

  Since φ_j holds in Σ_0, we have that

  $\hat{\Psi}(c, \sigma_0) = \hat{\Psi}(c^*, \sigma_0)$ (a4)

  From (a4) and (a1), we can conclude that

  $\hat{\Psi}(c, \sigma_0).2 = \Gamma_\Phi(c, \Sigma_0)$. (a5)

  From (a2), (a3), and the fact that φ_j holds in Σ_0, we can conclude that

  $\hat{\Psi}([c_1; \ldots; c_t], \sigma_0).2 = \Gamma_\Phi([c_1; \ldots; c_t], \Sigma_0)$ for $t = 1, \ldots, n - 1$. (a6)

  The inductive step for this case follows from (a5) and (a6).

- c_1 is a sequence of actions. Then, c_2 is a case plan. Assume that c_2 is the following case plan

  Case
  $\varphi_1 \rightarrow p_1$
  $\vdots$
  $\varphi_l \rightarrow p_l$
  Endcase

  Let $c^* = c_3; \ldots; c_n$. By definitions, we have that

  $$\Gamma_\Phi(c, \Sigma_0) = \Gamma_\Phi(c^*, \Gamma_\Phi(c_2, \Gamma_\Phi(c_1, \Sigma_0))). \tag{b1}$$

  Since $\perp \neq \Gamma_\Phi(c, \Sigma_0)$, we can conclude that there exists a j, $1 \leq j \leq l$, such that φ_j holds in $\Gamma_\Phi(c_1, \sigma_0)$. This implies that

  $$\Gamma_\Phi(c, \Sigma_0) = \Gamma_\Phi([c_1; p_j; c^*], \Sigma_0). \tag{b2}$$

  Since $c' = c_1; p_j; c^*$ is a conditional plan with $case(c') \leq k$, by induction hypothesis, there exists a rational model (σ_0, Ψ) of D satisfying the conditions of the Lemma w.r.t. c'. Similar to the first case, we can easily verify that (σ_0, Ψ) satisfies the lemma w.r.t. c. The inductive step is proved.

□

We now prove proposition 8.

Proposition 8 Let D be a consistent domain description, φ be a fluent formula in D, and c be a conditional plan in D. Then,

$$D \models^r_{\mathcal{A}_K} \varphi \textbf{ after } c \quad \text{iff} \quad D \models_{LTM} \varphi \textbf{ after } c.$$

Proof.

- Assume that $D \models^r_{\mathcal{A}_K} \varphi$ **after** c. We will prove that $D \models_{LTM} \varphi$ **after** c.

  Assume the contrary, i.e., there exists a LTM-model Φ of D such that φ does not hold in $\Gamma_\Phi(c, \Sigma_0)$. There are two cases:

  1. $\Gamma_\Phi(c, \Sigma_0) = \perp$. Then, by Lemma 4.31, there exists a rational model (σ_0, Ψ) of D such that $\hat{\Psi}(c, \sigma_0) = \perp$. This implies that $D \not\models^r_{\mathcal{A}_K} \varphi$ **after** c. This contradicts our assumption. So, this case cannot happen.
  2. $\Gamma_\Phi(c, \Sigma_0) \neq \perp$ and φ does not hold in $\Gamma_\Phi(c, \Sigma_0)$. Then, by Lemma 4.32, there exists a rational model (σ_0, Ψ) of D such that $\Gamma_\Phi(c, \Sigma_0) = \hat{\Psi}(c, \sigma_0).2$. This implies that φ does not hold in $\hat{\Psi}(c, \sigma_0)$. This implies that $D \not\models^r_{\mathcal{A}_K} \varphi$ **after** c. This contradicts our assumption. Hence, this case cannot happen too.

  From the above two cases, we can conclude that our that $D \models_{LTM} \varphi$ **after** c, i.e., we have proved that

  $$\text{if } D \models^r_{\mathcal{A}_K} \varphi \textbf{ after } c \text{ then } D \models_{LTM} \varphi \textbf{ after } c. \tag{a}$$

- Assume that $D \models_{LTM} \varphi$ **after** c. We will prove by contradiction that $D \models^r_{\mathcal{A}_K} \varphi$ **after** c. Assume that $D \not\models^r_{\mathcal{A}_K} \varphi$ **after** c. Then, by definition, there exists a rational model (σ_0, Ψ) of D such that φ does not hold in $\hat{\Psi}(c, \sigma_0)$. There are two cases:

  1. $\hat{\Psi}(c, \sigma_0) = \perp$. Then, by Lemma 4.31, there exists a LTM-model Φ of D such that $\Gamma_\Phi(c, \Sigma_0) = \perp$. This implies that $D \not\models_{LTM} \varphi$ **after** c. This contradicts our assumption. Hence, this case cannot happen.
  2. $\hat{\Psi}(c, \sigma_0) \neq \perp$ and φ does not hold in $\hat{\Psi}(c, \sigma_0)$. Then, by Lemma 4.33, there exists a LTM-model Φ of D such that $\Gamma_\Phi(c, \Sigma_0) = \hat{\Psi}(c, \sigma_0)$. This implies that φ does not hold in $\Gamma_\Phi(c, \Sigma_0)$, i.e., $D \not\models_{LTM} \varphi$ **after** c. This contradicts with our assumption, and hence, this case cannot happen too.

  The above two cases show that

  if $D \models_{LTM} \varphi$ **after** c then $D \models^r_{\mathcal{A}_K} \varphi$ **after** c. (b)

The proposition follows from (a) and (b). □

Appendix E

In this section, we will prove the consistency of the program $\pi_0(D)$ and the soundness and completeness of $\pi_0(D)$ w.r.t. the 0-Approximation of D. Like in Appendix C, we will assume that $\pi_0(D)$ consists of all ground instances of rules (3.1)-(3.6) where variables are instantiated using constants of the corresponding sorts. For an integer number i, by α_i we denote an arbitrary sequence of i actions of D, where $\alpha_0 = []$. We first prove the consistency of $\pi_0(D)$.

Proposition 10 (Consistency of $\pi_0(D)$) For every consistent domain description D, the program $\pi_0(D)$ is consistent.

Proof. Let $T = \{f \mid f$ is a fluent name, **initially** f is in $D\}$ and $F = \{f \mid f$ is a fluent name, **initially** $\neg f$ is in $D\}$.

It is easy to see that, for every i, the set

$$L_i = \{Holds(f, res(\alpha_j, s_0)) \mid j \leq i, \alpha_j \text{ is a sequence of } j \text{ actions, } f \text{ is a fluent literal}\} \cup \{ab(f, a, res(\alpha_j, s_0)) \mid j \leq i, \alpha_j \text{ is a sequence of } j \text{ actions, } f \text{ is a fluent literal, } a \text{ is an action}\}$$

is a splitting set of of $\pi_0(D)$. Since $lit(\pi_0(D)) = \bigcup_{0 \leq i} L_i$ and $L_j \subseteq L_i$ for $j < i$, $L = \langle L_i \rangle_{0 \leq i}$ is a splitting sequence of $\pi_0(D)$.

To prove the lemma we will prove that there exists a consistent solution $\langle X_i \rangle_{0 \leq i}$ to $\pi_0(D)$ with respect to L. To do that, we prove:

1. $b_{L_0}(\pi_0(D))$ has a consistent answer set X_0;
2. for every $i > 0$, $e_{L_i}(b_{L_{i+1}}(\pi_0(D)) \setminus b_{L_i}(\pi_0(D)), \bigcup_{j \leq i} X_j)$ has a consistent answer set X_{i+1}; and
3. $\bigcup_{0 \leq i+1} X_i$ is consistent.

Let $P_i = b_{L_i}(\pi_0(D))$.

Let us begin with item 1. By its construction, P_0 consists of rules of the forms (3.2), (3.3), and 3.5 of $\pi_0(D)$, which are:

$$
\begin{array}{rcl}
holds(f, s_0) & \leftarrow & (\text{if } \textbf{initially} \;\; f \in D) \\
\neg holds(f, s_0) & \leftarrow & (\text{if } \textbf{initially} \;\; \neg f \in D) \\
ab(\overline{f}, a, s) & \leftarrow & not \; \overline{Holds(p_1, s)}, \ldots, not \; \overline{Holds(p_m, s)} \\
 & & (\text{if } D \text{ contains } a \textbf{ causes } \; f \textbf{ if } \; p_1, \ldots, p_m)
\end{array}
$$

It is easy to check that the set

$$
\begin{array}{rl}
X_0 = & \{holds(f, s_0) \mid f \in T\} \cup \{\neg holds(f, s_0) \mid f \in F\} \cup \\
 & \{ab(f, a, s_0) \mid f \in F_a^-(\sigma_0)\} \cup \{ab(\overline{f}, a, s_0) \mid f \in F_a^+(\sigma_0)\}
\end{array}
$$

is a consistent answer set of P_0. Thus, item 1 is proved.

We now prove item 2 by induction over i. For the proof of the base case and the proof of the inductive step are similar, we omit here the proof of the base case.

Inductive step: Assume that we have proved that $e_{L_j}(P_{j+1} \setminus P_j, \bigcup_{k \leq j} X_k)$ has a consistent answer set X_{j+1} for every $j < i$. We need to prove that $e_{L_i}(P_{i+1} \setminus P_i, \bigcup_{k \leq i} X_k)$ has a consistent answer set X_{i+1}.

Let P be the program $e_{L_i}(P_{i+1} \setminus P_i, \bigcup_{k \leq i} X_k)$. By definition, P consists of rules of the following forms:

(a) $Holds(f, res(a, s)) \leftarrow$

   if there is a rule of the form (3.1) or (3.4) whose head is $Holds(f, res(a, s))$ and whose body is satisfied in $\bigcup_{k \leq i} X_k$;

(b) $ab(\overline{f}, a, s) \leftarrow not \; \overline{Holds(p_1, s)}, \ldots, not \; \overline{Holds(p_m, s)}$.

   if D contains an ef-proposition of the form

   a **causes** f **if** $p_1, \ldots, p_m$;

(c) $holds(f, res(a, s)) \vee \neg holds(f, res(a, s) \leftarrow$

   if there is a rule of the form (3.6) whose head is $holds(f, res(a, s)) \vee \neg holds(f, res(a, s)$ and whose body is satisfied in $\bigcup_{k \leq i} X_k$,

where s is of the form $res(\alpha_i, s_0)$ for some sequence of actions α_i.

We will prove that P is consistent. It is easy to see that it is enough to show that P does not contain $holds(f, res(a, s))$ and $\neg holds(f, res(a, s))$.

Assume the contrary, i.e., there exists a literal $holds(f, res(a, s)) \in P$ such that $\neg holds(f, res(a, s)) \in P$. From the construction of P, there are only three possibilities:

Case 1: $holds(f, res(a, s))$ and $\neg holds(f, res(a, s))$ are heads of rules of the form (3.1) of $\pi_0(D)$. This implies that $holds(f, s) \in \bigcup_{k \leq i} X_k$ and $\neg holds(f, s) \in \bigcup_{k \leq i} X_k$, which means that $\bigcup_{k \leq i} X_k$ is inconsistent. This contradicts the fact that $\bigcup_{k \leq i} X_k$ is consistent. Hence, this case cannot happen.

Case 2: $holds(f, res(a, s))$ is the head of a rule of the form (3.1) and $\neg holds(f, res(a, s))$ is the heads of a rule of the form (3.4) of $\pi_0(D)$. From the second assumption, we can conclude that there exists an ef-proposition of D of the form

a **causes** $\neg f$ **if** $p_1, \ldots, p_n$, and (b1)
$Holds(p_i, s) \in \bigcup_{k \leq i} X_k$ for $i = 1, \ldots, n$. (b2)

From (b2) and the assumption that $\bigcup_{k \leq i} X_k$ is consistent, we conclude that

$\overline{Holds(p_i, s)} \notin \bigcup_{k \leq i} X_k$ for $i = 1, \ldots, n$ (b3)

which implies that $ab(f, a, s) \in \bigcup_{k \leq i} X_k$, (b4)

which, in turn, implies that $holds(f, res(a, s)) \notin \bigcup_{k \leq i} X_k$. Thus, our assumption is incorrect, i.e., this case cannot happen too.

Case 3: $holds(f, res(a, s))$ and $\neg holds(f, res(a, s))$ are heads of rules of the form (3.4). By construction, this happens only if there exists two ef-propositions of the following forms:

a **causes** $\neg f$ **if** $p_1, \ldots, p_n$ and (c1)
a **causes** f **if** $q_1, \ldots, q_m$ (c2)

such that

$Holds(p_i, s) \in \bigcup_{k \leq i} X_k$ for $i = 1, \ldots, n$ and (c3)
$Holds(q_i, s) \in \bigcup_{k \leq i} X_k$ for $i = 1, \ldots, m$. (c4)

It is easy to see that the state $S = \{f \mid holds(f, s) \in \bigcup_{k \leq i} X_k\}$ satisfies both $\{p_1, \ldots, p_n\}$ and $\{q_1, \ldots, q_m\}$ and $\{p_1, \ldots, p_n\} \cap \{\bar{q}_1, \ldots, \bar{q}_m\} = \emptyset$. This contradicts the fact that D is a consistent domain description. Thus, this case cannot happen too.

The above three cases show that P is consistent. The inductive hypothesis is proved, i.e., we have proved that there exists a consistent solution $\langle X_i \rangle_{0 \leq i}$ of $\pi_0(D)$ with respect to $\langle L_i \rangle_{0 \leq i}$. From the splitting sequence theorem of [LT94] we can conclude that $\pi_0(D)$ has a consistent answer set. $\Box$

To prove the soundness and the completeness of $\pi_0(D)$ w.r.t. the 0-Approximation of D, we need the following lemmas.

Lemma 4.34 *Let D be a consistent domain description, σ_0 be its initial state, A be an answer set of $\pi_0(D)$ and α be a sequence of actions of D. Then, there exists an a-state $\langle T, F \rangle \in \hat{\Phi}(\alpha, \sigma_0)$ such that*

(i) $f \in T$ *iff* $holds(f, res(\alpha, s_0)) \in A$*; and*

(ii) $f \in F$ *iff* $\neg holds(f, res(\alpha, s_0)) \in A$.

Proof. We prove by induction over the length of α. Assume that $\sigma_0 = \langle T_0, F_0 \rangle$.

Base case: $|\alpha| = 0$. Then, we have that $\hat{\Phi}([], \sigma_0) = \sigma_0$. (a1)

Let A be an arbitrary answer set of $\pi_0(D)$. By construction of $\pi_0(D)$, we have that

if $f \in T_0$ then $holds(f, s_0) \in A$ and (by rule (3.2)) (a2) if $f \in F_0$ then $\neg holds(f, s_0) \in A$. (by rule (3.3)) (a3)

Now, assume that $holds(f, s_0) \in A$. Since $\pi_0(D)$ is consistent (Proposition 10), by Lemma 2.15, we can conclude that A is consistent. Hence, by Lemma 2.14, we have that there exists a rule in $\pi_0(D)$ whose head is $holds(f, s_0)$ and whose body is satisfied in A. From the construction of $\pi_0(D)$, this rule must be of the form (3.2), i.e., there exists a v-proposition **initially** f in D. Hence, $f \in T_0$. (a4)

Similarly, we can prove that if $\neg holds(f, s_0) \in A$ then $f \in F_0$. (a5)

The base case follows from (a1)-(a5).

Inductive step: Assume that we have proved the lemma for $|\alpha| = n$. We need to prove it for $|\alpha| = n + 1$. Assume that $\alpha = a_1, \ldots, a_{n+1}$. Let $\beta = a_1, \ldots, a_n$.

Let A be an answer set of $\pi_0(D)$ and

$T_i = \{f \mid holds(f, res([a_1, \ldots, a_i], s_0)) \in A\}$, and
$F_i = \{f \mid \neg holds(f, res([a_1, \ldots, a_i], s_0)) \in A\}$.

Let $Res(a_{n+1}, \langle T_n, F_n \rangle) = \langle T', F' \rangle$.

By inductive hypothesis, we have that $\langle T_n, F_n \rangle \in \hat{\Phi}(\beta, \sigma_0)$. Hence, to prove the inductive step, we will prove that

(*) $\langle T_{n+1}, F_{n+1} \rangle \in \Phi(a_{n+1}, \langle T_n, F_n \rangle)$.

First, we prove that $\langle T_{n+1}, F_{n+1} \rangle$ extends $\langle T', F' \rangle$. Assume that $f \in T'$. From the definition of Res, we have that

$f \notin F^-_{a_{n+1}}(\langle T_n, F_n \rangle)$ and (d1)

$f \in T \cup e^+_{a_{n+1}}(\langle T_n, F_n \rangle)$. (d2)

From (d1), we know that there exists no ef-proposition of the form a **causes** $\neg f$ **if** $p_1, \ldots, p_m$ such that $\{p_1, \ldots, p_n\}$ is possibly correct in $\langle T_n, F_n \rangle$. This, implies that, there exists some $p_k \in \{p_1, \ldots, p_m\}$ such that p_k is false in $\langle T_n, F_n \rangle$. Thus, $\overline{Holds(p_k, res(\beta, s_0))} \in A$. Hence, there exists no rule of the form

$$\neg holds(f, res(a_{n+1}, res(\beta, s_0))) \leftarrow Holds(p_1, res(\beta, s_0)), \ldots, Holds(p_n, res(\beta, s_0))$$

in $\pi_0(D)$ whose body is satisfied in A. And also, no rule of the form

$$ab(f, a_{n+1}, res(\beta, s_0)) \leftarrow not\ \overline{Holds(p_1, res(\beta, s_0))}, \ldots, not\ \overline{Holds(p_m, res(\beta, s_0))}$$

is applicable in A. Hence, by Lemma 2.14, we can conclude that

$ab(f, a_{n+1}, res(\beta, s_0)) \notin A$ and (d3)

$\neg holds(f, res(\alpha, s_0)) \notin A$. (d4)

Furthermore, from (d2) we have that one of the following cases holds.

Case 1: $f \in e^+_{a_{n+1}}$. Then, by definition, there exists an ef-proposition a **causes** f **if** $p_1, \ldots, p_m$ such that p_i holds in $\langle T_n, F_n \rangle$. Thus, by definition of T_n and F_n, $Holds(p_i, res(\beta, s_0)) \in A$. Therefore, there exists a rule of the form (3.2) in $\pi_0(D)$ whose head is $holds(f, res(\alpha, s_0))$ and whose body is satisfied in A. Hence, by Lemma 2.14, we conclude that $holds(f, res(\alpha, s_0)) \in A$, i.e., $f \in T_{n+1}$. (d5)

Case 2: $f \in T_n$. Then, we have that $holds(f, res(\beta, s_0)) \in A$. Together with (d3), we conclude that $holds(f, res(\alpha, s_0)) \in A$, i.e., $f \in T_{n+1}$. (d6)

It follows from (d5) and (d6) that if $f \in T'$ then $f \in T_{n+1}$. (d7)

Similarly, we can prove that if $f \in F'$ then $f \in F_{n+1}$. (d8)

From (d7) and (d8), we conclude that $\langle T_{n+1}, F_{n+1}\rangle$ extends $\langle T', F'\rangle$.

To complete the proof of (*) we need to show that

(**) $\langle T_{n+1}, F_{n+1}\rangle \setminus \langle T', F'\rangle = K(a_{n+1}, \langle T_n, F_n\rangle) \setminus \langle T', F'\rangle$.

Consider $f \in T_{n+1}$. By definition, we have that

$holds(f, res(\alpha, s_0)) \in A$ and (e1)
$f \notin T' \cup F'$. (e2)

Since A is an answer set of $\pi_0(D)$, there exists a rule in $\pi_0(D)$ whose head contains $holds(f, res(\alpha, s_0))$ and whose body is satisfied in A. Therefore, there are three cases:

Case 1: $holds(f, res(\alpha, s_0))$ is the head of a rule (3.2) of $\pi_0(D)$ whose body is satisfied in A. By construction of $\pi_0(D)$, there exists an ef-proposition

a_{n+1} **causes** f **if** $p_1, \ldots, p_m$ such that $Holds(p_i, res(\beta, s_0)) \in A$.

This implies that p_i holds in $\langle T_n, F_n\rangle$, and hence, $f \in e^+_{a_{n+1}}(\langle T_n, F_n\rangle)$. (e3)

Since D is consistent, similar to the proof of the consistency of $\pi_0(D)$, we can prove that $f \notin F^-_{a_{n+1}}(\langle T_n, F_n\rangle)$. (e4)

By definition, for this case, we have that $f \in T'$. (e5)

Case 2: None of the of the rule of form (3.2) whose head is $holds(f, res(\alpha, s_0))$ is applicable in A and $holds(f, res(\alpha, s_0))$ is the head of a rule (3.1) of $\pi_0(D)$ whose body is satisfied by A. This implies that

$holds(f, res(\beta, s_0)) \in A$ and (e6)
$ab(f, a_{n+1}, res(\beta, s_0)) \notin A$. (e7)

Then, from (e6), we conclude that $f \in T_n$ and from (e7) we have that there exists no ef-proposition of the form a_{n+1} **causes** f **if** $p_1, \ldots, p_m$ such that $\{p_1, \ldots, p_m\}$ is possibly true in T_n. This implies that $f \in T'$. (e8)

Case 3: There is no rule of the form (3.2) or (3.1) whose head is $holds(f, res(\alpha, s_0))$ whose body is satisfied by A. Then, $holds(f, res(\alpha, s_0))$ must belong to a rule (3.6) of $\pi_0(D)$. Thus, there exists a k-proposition a_{n+1} **determines** f. This implies that

$f \in K(a_{n+1}, \langle T_n, F_n\rangle)$. (e9)

It follows from (e8), (e9), and (e5) that

if $f \in T_{n+1}$ then $f \in T'$ or $f \in K(a_{n+1}, \langle T_n, F_n \rangle$. (e10)

Similarly, we can prove that

if $f \in F_{n+1}$ then $f \in F'$ or $f \in K(a_{n+1}, \langle T_n, F_n \rangle)$. (e11)

It follows from (e10) and (e11) that $\langle T_{n+1}, F_{n+1} \rangle \setminus \langle T', F' \rangle) \subseteq K(a_{n+1}, \langle T_n, F_n \rangle)$. (f1)

Now, assume that $f \in K(a_{n+1}, \langle T_n, F_n \rangle) \setminus \langle T', F' \rangle$. Then, by definition, we have that

there exists a k-proposition a_{n+1} **determines** f in D. Thus, the rule

$holds(f, res(\alpha, s_0)) \vee \neg holds(f, res(\alpha, s_0))$.

belongs to $\pi_0(D)$. Since A is an answer set of $\pi_0(D)$, we have that $holds(f, res(\alpha, s_0)) \in A$ or $\neg holds(f, res(\alpha, s_0)) \in A$. In both cases, we conclude that $f \in T_{n+1} \cup F_{n+1}$. Hence,
$K(a_{n+1}, \langle T_n, F_n \rangle) \setminus \langle T', F' \rangle \subseteq \langle T_{n+1}, F_{n+1} \rangle \setminus \langle T', F' \rangle)$. (f2)

(f1) and (f2) proves (**). Hence, we have that $\langle T_{n+1}, F_{n+1} \rangle \in \Phi(a_{n+1}, \langle T_n, F_n \rangle)$, i.e., (*) is proved. Thus, the inductive step the lemma is proved. □

The above lemma will help us to prove the completeness of $\pi_0(D)$ w.r.t. the 0-Approximation. We will now prove a lemma which will assist us in proving its soundness. Before doing that, let us introduce some notation.

Let us now fix a sequence of actions $\alpha = a_1, \ldots, a_n$. By γ_i, $1 \leq i \leq n$, we denote the sequence $a_1, \ldots, a_i$. Let $\gamma_0 = []$. We define a sequence of sets of literals U_i, $(0 \leq i \leq n)$, as follows.

$$\begin{aligned} U_i = \quad & \{Holds(f, res(\gamma_i, s_0)) \mid f \text{ is a fluent literal}\} \cup \\ & \{ab(f, a_{i+1}, res(\gamma_i, s_0)) \mid f \text{ is a fluent literal}\} \end{aligned}$$

for $0 \leq i \leq n$, where a_{n+1} is an arbitrary action in D.

It is easy to see that each of the set $S_j = \bigcup_{i=0}^{j} U_j$ is a splitting set of the program $\pi_0(D)$. Let P_j be the program consisting of rules whose head contains literals belonging to S_j. We prove

Lemma 4.35 *Let D be a consistent domain description and σ_0 be its initial state. Then, for every a-state $\langle T, F \rangle \in \hat{\Phi}([a_1, \ldots, a_j], \sigma_0)$, for $0 \leq j \leq n$, there exists an answer set A_j of P_j such that*

(i) $f \in T$ iff $holds(f, res(\gamma_j, s_0)) \in A_j$; and

(ii) $f \in F$ iff $\neg holds(f, res(\gamma_j, s_0)) \in A_j$

Proof. Let $\sigma_0 = \langle T_0, F_0 \rangle$. We prove the lemma by induction over j.

Base case: $j = 0$. It is easy to see that the set

$$\begin{aligned} A_0 = \quad & \{holds(f, s_0) \mid f \in T_0\} \cup \{\neg holds(f, s_0) \mid f \in F_0\} \cup \\ & \{ab(f, a_1, s_0) \mid f \in F^-_{a_1}(\sigma_0)\} \cup \{ab(\overline{f}, a_1, s_0) \mid f \in F^+_{a_1}(\sigma_0)\} \end{aligned}$$

is the unique answer set of P_0 that satisfies the conclusion of the lemma. The base case is proved.

Inductive step: Assume that we have prove the lemma for $j < n$. We need to prove it for $j + 1$. Let $\langle T, F \rangle$ be an arbitrary a-state in $\hat{\Phi}(\gamma_{j+1}, \sigma_0)$. Then, by definition, we conclude that there exists an a-state $\langle T', F' \rangle \in \hat{\Phi}(\gamma_j, \sigma_0)$ such that

$\langle T, F \rangle \in \Phi(a_{j+1}, \langle T', F' \rangle)$. (a)

By inductive hypothesis, we conclude that there exists an answer set A_j of P_j such that (b1) $f \in T'$ iff $holds(f, res(\gamma_j, s_0)) \in A_j$; and

(b2) $f \in F'$ iff $\neg holds(f, res(\gamma_j, s_0)) \in A_j$.

Since S_j is a splitting set of P_{j+1} and P_j is the bottom of P_{j+1} w.r.t. S_j, i.e., $P_j = b_{S_j}(P_{j+1})$, to prove the inductive step, it suffices to show that the partial evaluation of P_{j+1} w.r.t. S_j and A_j has an answer set A such that

(b3) $f \in T$ iff $holds(f, res(\gamma_{j+1}, s_0)) \in A$; and

(b4) $f \in F$ iff $\neg holds(f, res(\gamma_{j+1}, s_0)) \in A$.

By the splitting theorem, $P = e_{S_j}(P_{j+1} \setminus P_j, A_j)$ consists of following rules.

(c1) $Holds(f, res(\gamma_{j+1}, s_0)) \leftarrow$

if there is a rule of the form (3.4) or (3.1) whose head is $Holds(f, res(\gamma_{j+1}, s_0))$ and whose body is satisfied by A_j;

(c2) $ab(f, a_{j+1}, res(\gamma_j, s_0)) \leftarrow not\ \overline{Holds(p_1, res(\gamma_j, s_0))}, \ldots, not\ \overline{Holds(p_m, res(\gamma_j, s_0))}$

if there is an ef-proposition

a_{j+1} **causes** f **if** $p_1, \ldots, p_m$ in D; and

(c3) $holds(f, res(\gamma_{j+1}, s_0)) \vee \neg holds(f, res(\gamma_{j+1}, s_0)) \leftarrow$

if there is a k-proposition a_{j+1} **determines** f in D.

Let Q be the set of rules (c1) and (c2) of P. It is easy to see that the set U of literals formed by the predicate symbol $Holds$ is a splitting set of P and $b_U(P) = Q$. Notice also by the splitting theorem that each answer set A of P can be represented as a union of an answer set B of Q and an answer set C of the reduct of P w.r.t. U and B where $B \cap C = \emptyset$. Hence, it is enough to show that Q has an answer set B that satisfying (b3) and (b4) (where A is replaced by B in the conditions (b3) and (b4)). Let

$$B = \{holds(f, res(\gamma_{j+1}, s_0)) \mid f \in T\} \cup \{\neg holds(f, res(\gamma_{j+1}, s_0)) \mid f \in F\},$$

we will prove that B is an answer set of Q. Since Q does not contain negation-as-failure literals, we have that $Q^B = Q$. First, we prove that B satisfies the rules of Q.

Consider a rule r of the form $holds(f, res(\gamma_{j+1}, s_0)) \in Q^B$. Then, there are two cases:

Case 1: r is a rule of the form (3.4) whose head is $holds(f, res(\gamma_{j+1}, s_0))$ and whose body is satisfied by A_j. This implies that there exists an ef-proposition

a_{j+1} **causes** f **if** $p_1, \ldots, p_m$ in D and

$Holds(p_k, res(\gamma_j, s_0)) \in A_j$ for $1 \leq k \leq m$.

By inductive hypothesis, (b1), and (b2), we have that p_k $(1 \leq k \leq m)$ holds in $\langle T', F' \rangle$. Therefore, we can conclude that $f \in e^+_{a_{j+1}}(\langle T', F' \rangle)$. This implies that $f \in T$. (d1)

Case 2: r is a rule of the form (3.1). This implies that

$holds(f, res(\gamma_j, s_0)) \in A_j$ and (d2)

$ab(f, a_{j+1}, res(\gamma_j, s_0)) \notin A_j$. (d3)

It follows from (d2) and (b1) that $f \in T'$. (d4)

From (d3) we have that there exists no ef-proposition of the form

a_{j+1} **causes** $\neg f$ **if** $p_1, \dots, p_m$

such that for every $k = 1, \dots, m$, $\overline{Holds(p_i, res(\gamma_j, s_0))} \notin A_j$, i.e., for every ef-proposition a_{j+1} **causes** $\neg f$ **if** $p_1, \dots, p_m$ there exists some k, $1 \leq k \leq m$, such that $\overline{Holds(p_i, res(\gamma_j, s_0))} \in A_j$. Hence, by (b1) and (b2), we have that $f \notin F^{-}_{a_{j+1}}$. (d5)

From (d4) and (d5), we conclude that $f \in T'$. (d6)

From (d1) and (d6) we have that B satisfies rules of the type (c1) of Q^B. (d7)
Now, consider a rule r of type (c3) of Q^B. By definition, there exists a k-proposition

a_{j+1} **determines** f in D.

Hence, $f \in K(a_{j+1}, \langle T', F' \rangle)$. Here, we have two cases too.

Case 1: Either $holds(f, res(\gamma_{j+1}, s_0))$ or $\neg holds(f, res(\gamma_{j+1}, s_0))$ is the head of some rule of type (c1) of Q^B. Then, as we have proved above, either $holds(f, res(\gamma_{j+1}, s_0)) \in B$ or $\neg holds(f, res(\gamma_{j+1}, s_0)) \in B$, respectively. Hence, B satisfies r. (d8)

Case 2: neither $holds(f, res(\gamma_{j+1}, s_0))$ nor $\neg holds(f, res(\gamma_{j+1}, s_0))$ is the head of some rule of type (c1) of Q^B. Then, similar to the proof of (d1) and (d6), we can prove that f does not hold in $Res(a_{j+1}, \langle T', F' \rangle)$. This implies that $f \in K(a_{j+1}, \langle T', F' \rangle) \setminus Res(a_{j+1}, \langle T', F' \rangle)$, i.e., $f \in T \cup F$. Therefore, $holds(f, res(\gamma_{j+1}, s_0)) \in B$ or $\neg holds(f, res(\gamma_{j+1}, s_0)) \in B$. Hence, Hence, B satisfies r. (d9)

It follows from (d8) and (d9) that B satisfies rules of the type (c3) of Q^B. (d10)

From (d4) and (d10), we conclude that B satisfies rules of Q^B.

It remains to be shown that B is the minimal set of literals satisfying the rules of Q^B. Consider a fluent $f \in T$ (resp. $f \in F$). Since

$\langle T, F \rangle \setminus Res(a_{j+1}, \langle T', F' \rangle) = K(a_{j+1}, \langle T', F' \rangle) \setminus Res(a_{j+1}, \langle T', F' \rangle)$, we have that

$f \in T$ iff $f \in T' \cup e^{+}_{a_{j+1}}(\langle T', F' \rangle) \setminus F^{+}_{a_{j+1}}(\langle T', F' \rangle)$ or $f \in K(a_{j+1}, \langle T', F' \rangle)$.

In either cases, we can prove that there exists a rule in Q^B whose head is $holds(f, res(\gamma_{j+1}, s_0))$ (resp. $\neg holds(f, res(\gamma_{j+1}, s_0)))$ or $holds(f, res(\gamma_{j+1}, s_0)) \vee \neg holds(f, res(\gamma_{j+1}, s_0))$. This proves that B is the minimal set of literals satisfying the rules of Q^B, i.e., B is an answer set of P.

(b3) and (b4) follow from the construction of B. Hence, the inductive step is proved, and so, the lemma is proved. □

The next lemma follows immediately from the above lemma.

Lemma 4.36 *Let D be a consistent domain description, σ_0 be its initial state, and α be a sequence of actions of D. Then, for every a-state $\langle T, F\rangle \in \hat{\Phi}(\alpha, \sigma_0)$ there exists an answer set A of $\pi_0(D)$ such that*

(i) $f \in T$ *iff* $holds(f, res(\alpha, s_0)) \in A$; *and*

(ii) $f \in F$ *iff* $\neg holds(f, res(\alpha, s_0)) \in A$.

Proof. Assume that $\alpha = a_1, \ldots, a_n$. We have that S_n is a splitting set of $\pi_0(D)$ and the bottom program of $\pi_0(D)$ w.r.t. S_n is P_n. By Lemma 4.35, for every a-state $\langle T, F\rangle \in \hat{\Phi}(\alpha, \sigma_0)$ there exists an answer set A_n of P_n such that

- $f \in T$ iff $holds(f, res(\alpha, s_0)) \in A_n$; and
- $f \in F$ iff $\neg holds(f, res(\alpha, s_0)) \in A_n$.

By the splitting theorem, there exists a solution (A_n, B) of $\pi_0(D)$ w.r.t. S_n, i.e., $A = A_n \cup B$ is an answer set of $\pi_0(D)$. Furthermore, $A_n \cap B = \emptyset$. This implies that A is an answer set of $\pi_0(D)$ satisfying (i) and (ii). □

We are now ready to prove the Proposition 7.

Proposition 7 (Soundness and Completeness of $\pi_0(D)$ w.r.t. $\models_0$) Let D be a consistent domain description. Then, for every fluent literal f and sequence of actions α in D,

$$\pi_0(D) \models Holds(f, res(\alpha, s_0)) \quad \text{iff} \quad D \models_0 f \textbf{ after } \alpha.$$

Proof. We consider two cases:

1. Assume that $\pi_0(D) \models holds(f, res(\alpha, s_0))$. We will prove that $D \models_0 f$ **after** α. Assume the contrary, i.e., $D \not\models_0 f$ **after** α. This means that there exists an a-state $\langle T, F\rangle \in \hat{\Phi}(\alpha, \sigma_0)$ such that $f \notin T$. By Lemma 4.36, there exists an answer set A of $\pi_0(D)$ such that $holds(f, res(\alpha, s_0)) \notin A$. This contradicts the fact that $\pi_0(D) \models holds(f, res(\alpha, s_0))$. This means that our assumption is incorrect, i.e., we have proved that

   if $\pi_0(D) \models holds(f, res(\alpha, s_0))$ then $D \models_0 f$ **after** α. (a)

2. Assume that $D \models_0 f$ **after** α. We will prove that $\pi_0(D) \models holds(f, res(\alpha, s_0))$. Assume the contrary, i.e., $\pi_0(D) \not\models holds(f, res(\alpha, s_0))$. This means that there exists an answer set A of $\pi_0(D)$ such that $holds(f, res(\alpha, s_0)) \notin A$. By Lemma 4.34, there exists an a-state $\langle T, F\rangle \in \hat{\Phi}(\alpha, \sigma_0)$ such that $f \notin T$. This contradicts the fact that $D \models_0 f$ **after** α. This means that our assumption is incorrect, i.e., we have proved that

   if $D \models_0 f$ **after** α then $\pi_0(D) \models holds(f, res(\alpha, s_0))$. (b)

The proposition follows from (a) and (b). □