Formalizing sensing actions: a transition function based approach

Chitta Baral and Tran Cao Son
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
{chitta,tson}@cs.utep.edu

Abstract

In this paper we develop a high level action description language $\mathcal{A}_S$ that mirrors the solution to the frame problem for sensing actions in situation calculus developed by Scherl and Levesque. This is similar to the role the language $\mathcal{A}$ plays w.r.t. non-sensing actions. In defining the semantics of $\mathcal{A}_S$ we introduce the notion of a knowledge state which is a pair consisting of a state and a collection of states. The transition function of $\mathcal{A}_S$ is defined in such a way that it mirrors the successor state axiom for sensing actions by Scherl and Levesque. We then present a sound and complete translation of domains in $\mathcal{A}_S$ into disjunctive logic programs. Most importantly, using the new language we are able to prove the soundness of different approximation semantics of $\mathcal{A}_S$ that were developed by Baral and Son w.r.t. the semantics of Scherl and Levesque.

Introduction and Motivation

Reasoning about sensing actions (also called knowledge producing actions) is important[1] when planning in presence of incomplete information. It was first formally discussed in (Moore 1985), and later Scherl and Levesque (Scherl and Levesque 1993) gave successor state axioms and a solution of the frame problem in presence of sensing actions. Recently, Lobo et al (Lobo et al. 1997) present a high level action description language to represent and reason about sensing actions, and Baral and Son (Baral and Son, 1997) present an approximation semantics for reasoning about sensing actions.

In Scherl and Levesque's approach where the agent's knowledge about the world is formalized using the possible world models, a planner needs to keep track of the accessibility relations. For n fluents, in the worst case, the planner may have to keep track of 2^n possible worlds that are accessible. This increases the complexity of the planner tremendously. The same is true for Lobo et al's approach. In response to this Baral and Son present an approximation semantics which is weaker than both Scherl and Levesque's and Lobo et al's formalization but leads to a simpler (and faster) planner. We, therefore believe that most practical planner will follow the semantics of Baral and Son.

One of our main goal of this paper is to show that Baral and Son's semantics is sound w.r.t. Scherl and Levesque's formalization.

Besides, Scherl and Levesque's formalization given directly in classical logic is difficult to visualize and we believe it would be very useful to have a high level language with an automata-based semantics that corresponds to Scherl and Levesque's formalization. This language would play a similar role w.r.t. sensing actions, as the role the language $\mathcal{A}$ played w.r.t. non-sensing actions. Although, Lobo et al. do present a high level language for sensing actions, the relation between their language and that of Scherl and Levesque's formalization is not known. Moreover, Lobo et al. are only able to provide a translation of their formalism to epistemic logic programs which are more complex than disjunctive logic programs to which translation of $\mathcal{A}$ were provided.

In this paper we present a high level language for sensing actions 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. (Lobo et al. 1997). Moreover, we are also able to provide a sound and complete translation of descriptions in our language to disjunctive logic programs. Finally, using this high level language we are able to show the soundness of Baral and Son's approximation semantics w.r.t. Scherl and Levesque's formalization.

In our formalization, we separate the actual state of the world from the knowledge of an agent by introducing the notion of a *knowledge state* (or k-state). A k-state is a pair $\langle s, \Sigma \rangle$ where s is a state representing the actual state of the world and Σ is a collection of states representing the set of possible states which an agent

[1] Due to space limitations we only cite a few of the papers dealing with planning in presence of sensing actions and incomplete information.

believes he may be in. The semantics of $\mathcal{A}_S$ is then defined by transition functions which map pairs of actions and k-states into k-states.

The language $\mathcal{A}_S$

In this section we introduce a variation of the language $\mathcal{A}$ in (Gelfond and Lifschitz, 1993), $\mathcal{A}_S$, which allows reasoning about sensing actions.

Syntax of $\mathcal{A}_S$

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 $\overline{\neg f}$ and $\overline{f}$ we mean f and $\neg f$ respectively.

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

$$\textbf{initially} \quad f, \tag{1.1}$$

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

$$a \ \textbf{causes} \ \ f \ \textbf{if} \ \ p_1, \ldots, p_n \tag{1.2}$$

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

$$a \ \textbf{determines} \ p \tag{1.3}$$

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, ex-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.

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

Queries in $\mathcal{A}_S$ 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
   $\varphi_1 \rightarrow c_1$
   ...
   $\varphi_n \rightarrow c_n$
   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 w.r.t. 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 is known to be 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:

$$\phi \text{ after } c \tag{1.4}$$

$$\textbf{Kwhether } \phi \textbf{ after } c \tag{1.5}$$

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

Semantics of $\mathcal{A}_S$

In $\mathcal{A}_S$, 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 $\mathcal{A}_S$ is 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 $p_1, \ldots, p_n$ are satisfied 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 $p_1, \ldots, p_n$ are satisfied in $s\}$.

The transition function of D is defined next.

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, $\Phi(a, \sigma) = \langle Res(a, s), \{s' \mid s' = Res(a, s'')$ 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** $f_1, \ldots,$ a **determines** f_m, $\Phi(a, \sigma) = \langle s, \{s' \mid s' \in \Sigma$ such that s and s' agree on f_i $(i = 1, \ldots, m)\}\rangle$

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.* 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 it 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 a case plan c, defined as in Item 3 of Definition 1,

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

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.

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

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

(i) $D \models_{\mathcal{A}_S} \varphi$ **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) $D \models_{\mathcal{A}_S}$ **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 $\models^r_{\mathcal{A}_S}$ – is defined similarly by only considering rational models of D. □

We illustrate our definitions in the next example.

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_1: $\sigma = \langle s_1, \{s_1, s_2\}\rangle$ and $\sigma' = \langle s_2, \{s_1, s_2\}\rangle$. We have that

$\hat{\Phi}([a], \sigma) = \langle s_3, \{s_3, s_4\}\rangle$,
$\hat{\Phi}([a, sense_g], \sigma) = \hat{\Phi}([sense_g], \langle s_3, \{s_3, s_4\}\rangle) = \langle s_3, \{s_3\}\rangle$

and

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

We depict the result of Φ on two k-states in picture 1.

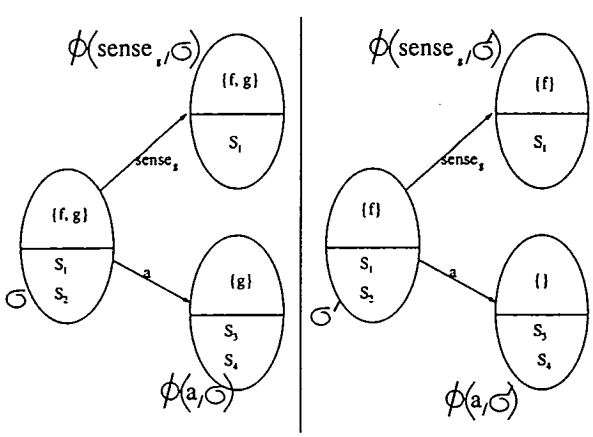

Figure 1: The result of Φ on σ (left) and σ' (right).

Since g is known to be true in $\langle s_3, \{s_3\}\rangle$ and known to be false in $\langle s_4, \{s_4\}\rangle$, we can conclude that $D_2 \models^r_{\mathcal{A}_S}$ **Kwhether** (g) **after** $[a, sense_g]$.

However, $D_2 \not\models^r_{\mathcal{A}_S}$ **Kwhether** (g) **after** $[a]$, $D_2 \not\models^r_{\mathcal{A}_S}$ g **after** $[a, sense_g]$, and $D_2 \not\models^r_{\mathcal{A}_S}$ $\neg g$ **after** $[a, sense_g]$.

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

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

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.

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

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

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

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

and

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

It is easy to see that for every sequence of actions α of D_1, $D_1 \not\models^r_{\mathcal{A}_K}$ *disarmed*$\neg$*exploded* **after** α. It is, however, easy to check that $D_1 \models^r_{\mathcal{A}_S}$ *disarmed* $\wedge$ $\neg$*exploded* **after** c where c is the following conditional plan.

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

Translation into Logic Program

We now present a translation from domain description in $\mathcal{A}_S$ into disjunctive logic programs and show that the translation (using answer-set semantics (Gelfond and Lifschitz, 1991)) is sound and complete w.r.t. the semantics of $\mathcal{A}_S$. *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 constant start with lower case letters.

The translation $\pi(D)$ of a domain description D, uses variables of three sorts: *state* 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 $H(f, s)$ we denote $h(f, s)$ and by $H(\bar{f}, s)$ we denote $\neg h(f, s)$. Also, for a fluent f, by $\bar{f}$ we denote $\neg f$ and by $\neg f$ 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, $h(F, S)$ (resp. $\neg h(F, S)$) means that the fluent F is true (resp. false) in the situation S. A conditional plan is represented here as a list of pairs of conditions and plans together with a function symbol 'case' that is used as a constructor. For example, the conditional plan c in the Example 3 is represented by the list $[look,\ case([([\neg locked], [turn]), ([locked], [])]),\ disarm]$. 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 (1.1), (1.2) and (1.3), respectively, are represented by the following rules:

$$initially(f). \qquad (\pi.1)$$

$$causes(a, f, [p_1, \ldots, p_n]). \qquad (\pi.2)$$

$$determines(a, f). \qquad (\pi.3)$$

- **Domain independent rules**:
  - **The inertial rule**

$$H(F, res(A, S)) \leftarrow H(F, S), not\ ab(F, A, S). \qquad (\pi.4)$$

  - **Rules describing effects of actions:**

$$hl([], S) \leftarrow \qquad (\pi.5)$$

$$hl([H|T], S) \leftarrow H(H, S), hl(T, S). \qquad (\pi.6)$$

$$H(F, res(A, S)) \leftarrow causes(A, F, L), hl(L, S). \qquad (\pi.7)$$

$$ab(\bar{F}, A, S) \leftarrow causes(A, F, L), hl(L, S). \qquad (\pi.8)$$

$$acc(res(A, S_2), res(A, S_1)) \leftarrow acc(S_2, S_1), causes(A, F, L). \qquad (\pi.9)$$

$$\neg acc(res(A, S_2), res(A, S_1)) \leftarrow acc(S_2, S_1), determines(A, F), \neg agree(F, S_1, S_2). \qquad (\pi.10)$$

$$acc(res(A, S_2), res(A, S_1)) \leftarrow acc(S_2, S_1), determines(A, F), not\ \neg acc(res(A, S_2), res(A, S_1)). \qquad (\pi.11)$$

Rules in this group describe the effects of actions. The first two rules define hl in terms of h, 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:**

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

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

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

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

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

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 $h(F, s_0)$ ($\neg h(F, s_0)$) holds. Rule (π.14) indicates that for every fluent F, either $H(F, s_0)$ or $H(\bar{F}, s_0)$ holds. 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:**

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

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

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 (1.4) and (1.5).

$$apply([], S, S) \leftarrow \quad (\pi.19)$$

$$apply([A|T], S, S1) \leftarrow action(A), apply(T, res(A, S), S1). \quad (\pi.20)$$

$$\neg k(F, S_1) \leftarrow acc(S_2, S_1), H(\bar{F}, S_2). \quad (\pi.21)$$

$$k(F, S) \leftarrow not\ \neg k(F, S). \quad (\pi.22)$$

$$kw(F, S) \leftarrow k(F, S). \quad (\pi.23)$$

$$kw(F, S) \leftarrow k(\bar{F}, S). \quad (\pi.24)$$

$$knows(F, C, S) \leftarrow apply(C, S, S1), S1 \neq \perp, k(F, S1). \quad (\pi.25)$$

$$klist([], S) \leftarrow \quad (\pi.26)$$

$$klist([H|T], S) \leftarrow k(H, S), klist(T, S). \quad (\pi.27)$$

$$apply(X, \perp, \perp) \leftarrow \quad (\pi.28)$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow klist(C, S), apply(P, S, S1), apply(L, S1, X). \quad (\pi.29)$$

$$apply([case([(C, P)|T])|L], S, X) \leftarrow not\ klist(C, S), apply([case(T)|L], S, X). \quad (\pi.30)$$

$$apply([case([])|L], S, \perp) \leftarrow \quad (\pi.31)$$

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.

- <u>**Language dependent rules:**</u>
  - **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:

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

  - **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 $\mathcal{A}_S$-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.

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

Rules enumerating the set of possible states for D_2 are:

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

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

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

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

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

The next proposition shows that $\pi(D)$ is sound and complete w.r.t. the $\models^r_{\mathcal{A}_S}$.

Proposition 1 (Soundness and Completeness) Let D be a consistent domain description and f be a fluent in D. Then, for any conditional plan c, $\pi(D) \models knows(f, c, s_0)$ iff $D \models^r_{\mathcal{A}_S} f$ **after** c.

Related Research

In this section, we discuss the relationship between our approach and the two most recent formalisms on formalizing sensing actions, one by Scherl and Levesque (Scherl and Levesque 1993) and the other by Lobo et al. (Lobo et al. 1997).

Relationship with Scherl and Levesque's formalism

In (Scherl and Levesque 1993), Scherl and Levesque extend Reiter's solution to the frame problem for non-sensing actions (Reiter 1991) and Moore's work on knowledge and action (Moore 1985) to formulate reasoning with knowledge producing actions. We will show next that each domain description in $\mathcal{A}_S$ can be translated into a first order theory that uses Scherl and Levesque successor state axiom for sensing actions and show the equivalence w.r.t. queries in $\mathcal{A}_S$. To do that, we translate each domain D into an equivalent domain in situation calculus using the following steps:

1. Use the translation in (Kartha 1995) to translate the non-sensing part of D into an equivalent situation calculus domain, denoted by $R_1(D)$.
2. Let **initially** G_i $(i = 1, \ldots, k)$ be the k v-propositions of D. The following axiom is for the accessible relation in the initial situation:

$$K(s, S_0) \supset \bigwedge_{i=1}^{k} G_i(s). \qquad (4.1)$$

$$K(S_0, S_0) \qquad (4.2)$$

3. If D contains n k-propositions A_i **determines** F_i $(i = 1, \ldots, n)$, the successor state axiom for K is defined as follows.

$$\begin{aligned}&[K(s'', do(x, s)) \equiv \\ &\quad \exists s'.((K(s', s) \wedge s'' = do(x, s'))\wedge \\ &\quad (\bigwedge_{j=1}^{n}(x \neq A_j)\wedge \\ &\quad \bigvee_{j=1}^{n}(x = A_j \wedge f_j(s) \equiv f(s'))))] \quad (4.3)\end{aligned}$$

Finally, we add the axioms (4.1), (4.2), and (4.3) to $R_1(D)$. We call the resultant theory $R(D)$. The relationship between the entailments in $R(D)$ and in D is stated in the next proposition.

Proposition 2 Let D be a consistent domain description in $\mathcal{A}_S$ and $R(D)$ be its situation calculus counterpart. Then, for any fluent formula φ and sequence of actions α, $D \models \varphi$ **after** α iff $R(D) \models Knows(\varphi, do(\alpha, S_0))$ where $Knows(\varphi, S)$ is a shorthand for $\forall s'.(K(s', S) \supset \varphi(s'))$.

We also investigate the relationship between 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 (Levesque 1996).) We define *Apply* as a nested abnormality theory (NAT) block.

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

We then prove that

Proposition 3 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, $D \models \varphi$ **after** c iff $R(D) \cup B_{\text{Apply}} \models Apply(c, S_0, s) \wedge Knows(\varphi, s)$ where $Knows(\varphi, S)$ is a shorthand for $S \neq \bot \wedge \forall s'(K(s', S) \supset \varphi(s'))$.

Relationship with Lobo et al.'s semantics

In (Lobo et al. 1997), $\mathcal{A}_K$, an extension of $\mathcal{A}$ has been introduced to allow reasoning about sensing actions. The syntax of $\mathcal{A}_K$ and $\mathcal{A}_S$ are almost identical. The differences between the two approaches are discussed in details in (Baral and Son, 1998). We prove that

Proposition 4 Let D be a domain description, φ be a fluent formula in D, and c be a conditional plan in D. Then, $D \models^r_{\mathcal{A}_S} \varphi$ **after** c iff $D \models_{LTM} \varphi$ **after** c where $\models_{LTM}$ denotes the entailment relationship defined by the semantics of (Lobo et al. 1997).

Approximation Semantics

In (Baral and Son, 1997), Baral and Son proposed two approximation semantics for $\mathcal{A}_S$. In their formulation, a state of the world (from the agent's point of view) is represented as a 3-valued interpretation of the fluents in the world and is denoted by a pair $\langle T, F\rangle$, where T is the set of fluents that have truth value *true*, F is the set of fluents that have truth value *false*, and T and F are disjoint. The rest of the fluents in the world have truth value *unknown*. I.e., the agent does not know if those fluents are true or not. Sensing actions are formalized by defining a transition function Φ such that if a is an action that senses g and $g \notin T \cup F$, then $\Phi(\langle T, F\rangle)$ is the set of states $\{\langle T \cup \{g\}, F\rangle, \langle T, F \cup \{g\}\rangle\}$.

This is more general than the formulation in $\mathcal{A}$, where states are 2-valued interpretation of the fluents in the world but it is less general $\mathcal{A}_S$ where we use k-states. For a domain of n fluents, there are only 3^n possible 3-valued states whereas there are $2^n \times 2^{2^n}$ possible k-states. Their semantics is simpler than the semantics of $\mathcal{A}_S$ presented in this paper.

Although it is known that their semantics is weaker than the semantics of Scherl and Levesque (and hence weaker than the semantics of $\mathcal{A}_S$), we can show the soundness of their semantics w.r.t. the semantics of

Scherl and Levesque. This follows from the following proposition and Proposition 2.

Proposition 5 Let D be a consistent domain description in $\mathcal{A}_S$. Then, for any fluent f and sequence of action α, if $D \models f$ **after** α (w.r.t. the approximation semantics of Baral and Son) then $D \models_{\mathcal{A}_S} f$ **after** α.

Conclusion and Future Work

In this paper we developed a high level action description language that allows sensing actions. We also provide a sound and complete translation of domain descriptions in $\mathcal{A}_S$ into disjunctive logic programs. To the best of our knowledge, this is the first sound and complete translation of domains in a dialect of $\mathcal{A}$ that allows sensing actions into disjunctive logic programs. (Lobo et al's translation was to epistemic logic programs which is more powerful than disjunctive logic programs.)

We show the relation between our high level language and other formalisms such as those of Scherl and Levesque and Lobo et. al. We also prove the soundness of Baral and Son's approximation semantics w.r.t. our new formalism.

In the near future we would like to develop a planner which uses our semantics in generating and verifying conditional plans.

References

[Baral and Son, 1997] Baral, C. and Son T.C.: *Approximate Reasoning about Actions in Presence of Sensing and Incomplete Information.* ILPS'97, 387–401.

[Baral and Son, 1998] Baral, C. and Son T.C.: *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.

[Gelfond and Lifschitz, 1991] Gelfond, M., Lifschitz, V.: *Classical Negation in Logic Programs and Disjunctive Databases*, New Generation of Computing 365–387, 1991.

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

[Kartha 1995] Kartha, G.N.: *A mathematical investigation of reasoning about actions.* Ph.D dissertation. University of Texas at Austin, 1995.

[Levesque 1996] Levesque, H.J.: *What is planning in the presence of sensing*, in Proc. of AAAI, 1996, 1139-1146.

[Lobo et al. 1997] Lobo, J., Taylor, S., and Mendez, G.: *Adding knowledge to the action description language* $\mathcal{A}$, in Proc. of AAAI 97, 454–459.

[Moore 1985] Moore, R.C.. *A formal theory of knowledge and action,* in: Hobs, J.R. and Moore, R.C. eds. 1985. Formal Theories of the Commonsense World, 319–358.

[Reiter 1991] Reiter, R.: *The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression,* in: V. Lifschitz (ed.): AI and Mathematical Theory of Computation: Papers in Honor of John McCarthy, Academic Press, San Diego, 1991, 359–380.

[Scherl and Levesque, 1993] Scherl, R., Levesque, H.J.: *Knowledge Producing Actions*, In Proc. of KR, 1993, 1139-1146.