On the Completeness of Approximation Based Reasoning and Planning in Action Theories with Incomplete Information

Tran Cao Son and **Phan Huy Tu**
Computer Science Department
New Mexico State University
PO Box 30001, MSC CS
Las Cruces, NM 88003, USA
tson|tphan@cs.nmsu.edu

Abstract

In this paper, we study the completeness of the 0-approximation for action theories with incomplete information. We propose a sufficient condition for which an action theory under the 0-approximation semantics is complete with respect to the possible world semantics. We then introduce the notion of *decisive sets of fluents*, based on which an action theory can be modified into another action theory such that the modified action theory under the 0-approximation is complete with respect to the original theory. We present a polynomial time algorithm for computing decisive sets for action theories and use it in the development of a sound and complete conformant planner. Finally, we compare our planner with other state-of-the-art conformant planners.

Introduction

Reasoning about effects of actions in the presence of incomplete information has been widely studied by AI researchers (Etzioni *et al.* 1992; Golden, Etzioni, & Weld 1996; Golden & Weld 1996; Goldman & Boddy 1994; 1996; Levesque 1996; Moore 1985; Peot & Smith 1992; Pryor & Collins 1996; Smith & Weld 1998; Thielscher 2002; Weld, Anderson, & Smith 1998; Cimatti, Roveri, & Bertoli 2004). Most of early proposals rely on the possible world semantics that is introduced in (Moore 1985). The basic idea of this approach lies in that to reason about the effects of an action (or an action sequence) with its incomplete knowledge about the current state of the world, an agent has to consider *all possible state of the worlds* which are consistent with its knowledge. Following this approach, the problem of finding a (polynomial length) conformant plan is $\Sigma_2\mathbf{P}$-complete (Baral, Kreinovich, & Trejo 2000).

An alternative alternative to the possible world semantics is based on approximations (Son & Baral 2001). The basic idea is to approximate the set of possible world states by a single partial state. Perhaps the main advantage of the approximation-based approach is its low complexity in reasoning and planning tasks (Baral, Kreinovich, & Trejo 2000). It has proved to be useful in the development of a regression based conditional planner (Tuan *et al.* 2004) and a logic programming based conditional planner (Son, Tu, & Baral 2004). It has also been extended to action theories with state constraints and used in the implementation of different conformant planners whose performance is comparable to those of state-of-the-art conformant planners (Son *et al.* 2005b; 2005a).

The main weakness of the approximation-based approach is its incompleteness[1] w.r.t. the possible world semantics, i.e., a reasoner based on this approach may answer a query about the truth value of a fluent formula after the execution of a sequence of actions with 'unknown' while another reasoner using the possible world approach would answer with either 'true' or 'false'. This also implies that a conformant planner based on approximations may not find a plan even when one exists. In this paper, we investigate methods that allow for complete reasoning using one of those approximations, called the 0-approximation (Son & Baral 2001).

A trivial method is to do exactly what the possible world approach does: considers *all* possible initial states. This solution is *not satisfactory* since *(i)* it does not scale up well; and *(ii)* in several cases, it is *not necessary* as can be seen in the following example.

Example 1 Consider a simple instance of the bomb-in-the-toilet domain with only one toilet and one package. Initially, the package may or may not contain a bomb and whether or not the toilet is clogged is unknown. Dunking the package into the toilet disarms the bomb; in addition, it also makes the toilet clogged. This action can be executed only if the toilet is unclogged. Flushing the toilet makes it unclogged. One can encode this domain in the language $\mathcal{A}$ (Gelfond & Lifschitz 1992) as follows.

$$\mathcal{D}_1 = \left\{ \begin{array}{l} dunk \textbf{ causes } \neg armed \textbf{ if } armed \\ dunk \textbf{ causes } clogged \\ flush \textbf{ causes } \neg clogged \\ \textbf{executable } dunk \textbf{ if } \neg clogged \end{array} \right\}$$

Intuitively, we would like to conclude that the bomb will be disarmed after the execution of the sequence of actions $[flush; dunk]$ no matter what the initial state is. Figures 1a-b illustrate the reasoning process based on the possible world approach and on the 0-approximation respectively.

Using the possible world approach (Figure 1a), since both $armed$ and $clogged$ are unknown in the beginning,

[1] Soundness of the approximations is proved in (Son & Baral 2001).

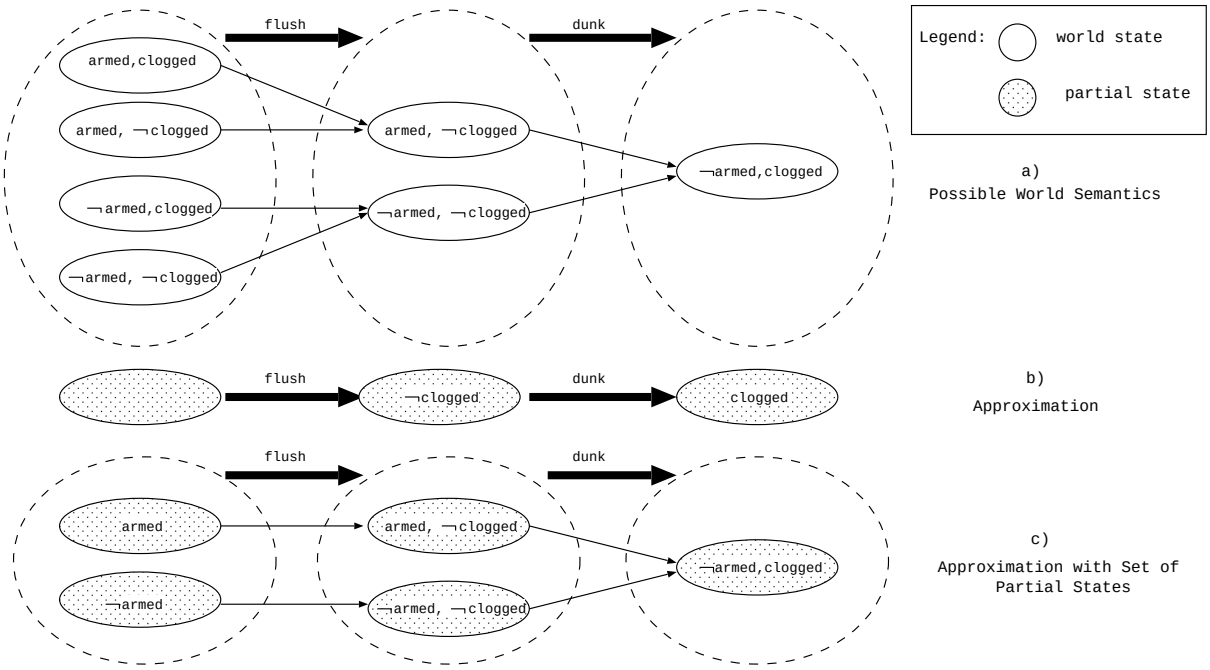

Figure 1: Possible World Semantics vs. Approximation Based Reasoning

there are four possible initial states corresponding to different assignments of their truth values. After performing the action $flush$, the two possible successor states are $\{armed, \neg clogged\}$ and $\{\neg armed, \neg clogged\}$ as flushing the toilet makes it unclogged. After performing $dunk$, the final state of the world is $\{\neg armed, clogged\}$. Consequently, we can conclude that the bomb would be disarmed in the final state.

The 0-approximation (Figure 1b), on the other hand, does not help us to draw that conclusion. The reason is that it approximates the four possible initial states by $\emptyset$, the empty partial state[2], and performing the action $flush$ in $\emptyset$ results in the partial state $\{\neg clogged\}$. Executing $dunk$ in this partial state results in the final partial state $\{clogged\}$ and thus $armed$ is unknown.

It is easy to see that if the truth values of $armed$ were considered separately in the beginning (by partitioning the empty partial state into two possible partial states $\{armed\}$ and $\{\neg armed\}$) then the status of $armed$ after the execution of $[flush; dunk]$ will be known (see Figure 1c). Partitioning the initial partial state over $\{clogged\}$, on the contrary, does not help us to draw the conclusion. □

Similar situations may happen when executability conditions of actions are taken into account. The following example illustrates this point.

Example 2 Consider the domain:

$$\mathcal{D}_2 = \left\{ \begin{array}{l} a \textbf{ causes } g, \\ \textbf{executable } a \textbf{ if } h \\ \textbf{executable } a \textbf{ if } \neg h \end{array} \right\}$$

[2]A partial state is a consistent set of fluent literals.

Assume that we know nothing about the initial state of the world. It is easy to check that the 0-approximation will not allow us to conclude that g is true after the execution of a. Nevertheless, if we consider $\{h\}$ and $\{\neg h\}$, instead of $\emptyset$, as the two possible initial partial states then the 0-approximation will allow us to conclude that g is true after the execution of a. □

The above examples show that the completeness of reasoning based on the 0-approximation (w.r.t. the possible world semantics)[3] sometimes can be achieved without having to examine all possible initial states. It also raises the following question: why did we choose the set of fluents $\{armed\}$ but not $\{clogged\}$ in Example 1 (or $\{h\}$ but not $\{g\}$ in Example 2) to partition the initial knowledge? In a more general form, the question is: what fluents should be chosen to split the initial knowledge in order for the 0-approximation to be complete. One of our main goals in this paper is to address the problem of identifying such fluents. Furthermore, the chosen set of fluents should be as small as possible because it helps reduce the number of possible initial partial states and gain in efficiency.

Our approach to solve this problem as follows. First, we study a sufficient condition for which an action theory[4] with a single initial partial state under the 0-approximation semantics is complete. Second, we introduce a notion called a *decisive set* for action theories with a single partial state δ. Intuitively, a decisive set is a set F of unknown fluents

[3]From now on, whenever we say about the incompleteness or completeness of an approximation, we mean it with respect to the possible world semantics

[4]An action theory is a domain description and a set of partial states describing the initial knowledge of an agent about the world.

such that the 0-approximation is complete if the partition of δ using F is considered. This is then extended to the case when the initial state of the world is described by more than one partial state (i.e., disjunctive information about the initial state is included). To evaluate the usefulness of our proposal, we develop a sound and complete conformant planning system based on this idea, called CPA$^+$. The experiments show that CPA$^+$ is competitive with other state-of-the-art planners.

To summarize, the main contributions of the paper are:

- A sufficient condition for which an action theory is complete (Theorem 1).
- The notion of decisive sets (Definition 4); these sets can be used to modify an action theory in such a way that the 0-approximation semantics of the modified theory is complete with respect to the possible world semantics of the original theory (Theorems 2 and 3).
- A polynomial time algorithm for computing decisive sets (Figure 2).
- A sound and complete planning system based on the 0-approximation that exploits the above results; its performance is shown to be competitive with other state-of-the-art conformant planners.

The paper is organized as follows. In Section 2, we review the basics of an action language, the possible world semantics, and the 0-approximation semantics. In Section 3, we present our proposal to make the 0-approximation complete. Section 4 shows how our study on the completeness of the 0-approximation can be used to develop a sound and complete conformant planner and presents some experimental results. Sections 5 and 6 provide discussion and related work, and the conclusion of the paper.

Preliminaries

In this section, we first review the basic definitions of a variant of the language $\mathcal{A}$ from (Gelfond & Lifschitz 1992) that allows for the representation of actions with conditional effects and executability conditions. We then review the possible world semantics and the 0-approximation.

Action Representation

The alphabet of a domain consists of a set **A** of action names and a set **F** of fluent names. A (fluent) literal is either a fluent $f \in \mathbf{F}$ or its negation $\neg f$. A *fluent formula* is composed of literals and connectives $\wedge$, $\vee$, and $\neg$ as usual. A *domain description* (or a domain for short) $\mathcal{D}$ is a set of laws of the following forms:

$$a \ \textbf{causes} \ l \ \textbf{if} \ \psi \tag{1}$$

$$\textbf{executable} \ a \ \textbf{if} \ \psi \tag{2}$$

where $a \in \mathbf{A}$ is an action, l is a literal, and ψ is a set of literals. (1) is called a *dynamic law*, describing the effect of action a. It says that if a is performed in a situation wherein ψ holds then l will hold in the successor situation. (2) is an *executability condition* on a, stating that a is executable in any situation in which ψ holds.

For a literal l, by $\neg l$ we denote its complement. For a set σ of literals, we denote by $\neg\sigma$ the set $\{\neg l \mid l \in \sigma\}$. σ is consistent if it does not contain two contrary literals, that is, for every literal l, either l or $\neg l$ does not belong to σ. In this paper, we will use two terms *consistent set of literals* and *partial state* alternatively, depending on the context they are being used. σ is complete if for every fluent f, either f or $\neg f$ belongs to σ. When σ is consistent and complete, it is called a *state*.

Given a consistent set σ of literals, a literal l (resp. set of literals γ) *holds* in a set of literals σ if $l \in \sigma$ (resp. $\gamma \subseteq \sigma$); l (resp. γ) *possibly holds* in σ if $\neg l \notin \sigma$ (resp. $\neg\gamma \cap \sigma = \emptyset$). The value of a formula φ in σ, denoted by $\sigma(\varphi)$, may be either true, false, or unknown and is defined as usual. It is easy to see that if σ is a state then for every formula φ, the value of φ is known in σ. An action a is *executable* in σ if there exists an executability condition (2) on a such that ψ holds in σ; a is executable in a set Σ of consistent sets of literals if a is executable in every element of Σ. From now on, to avoid confusion, we will use letters (possibly indexed) σ, δ, and s to denote a set of literals, a partial state, and a state respectively.

Given a domain $\mathcal{D}$, for a state s and an action a executable in s, the direct effect of a in s is defined by

$$e(a, s) = \{l \mid a \ \textbf{causes} \ l \ \textbf{if} \ \psi \in \mathcal{D}, \psi \text{ holds in } s\}$$

$\mathcal{D}$ is *inconsistent* if there exist state s and action a executable in s such that $e(a, s)$ is inconsistent. In the rest of the paper, we are interested in consistent domains only. The successor state after executing a in s, $Res^c(a, s)$, is defined as follows.

$$Res^c(a, s) = \begin{cases} (s \cup e(a, s)) \setminus \neg e(a, s) & \\ \qquad\qquad \text{if } a \text{ is executable in } s \\ \bot \qquad\qquad \text{otherwise} \end{cases}$$

where $\bot$ denotes that the execution of a in s fails. For convenience, we sometimes use the notation $Res^c(a, S)$, where S is a set of states, to refer to $\bigcup_{s \in S}\{Res(a, s)\}$.

It can be proved that if $\mathcal{D}$ is consistent and a is executable in s then $Res^c(a, s)$ is a state. The Res^c-function is then extended for reasoning about effects of a sequence of actions as follows.

$$\Phi^c([a_1; \ldots; a_n], s) = \begin{cases} \bot & \text{if } s = \bot \\ s & \text{if } n = 0 \\ \Phi^c([a_2; \ldots; a_n], Res^c(a_1, s)) & \\ \qquad \text{if } n \geq 1 \end{cases} \tag{3}$$

This function can be used to answer queries of the form

$$\varphi \ \textbf{after} \ \alpha, \tag{4}$$

where α is an action sequence and φ is a formula. It asks whether φ is true in the final state after the execution of α in the initial state.

In the presence of incomplete information about the initial state, the initial state is not completely specified. In general, an *action theory* is given by a pair $(\mathcal{D}, \Delta)$ where $\mathcal{D}$ is a domain and Δ is a non-empty set of partial states representing the initial state [5]. Queries (4) can be answered by using the

[5] This allows for an explicit representation of disjunctive information about the initial state. When no disjunctive information about the initial state is available Δ is a singleton, i.e., $|\Delta| = 1$.

possible world semantics (Moore 1985). Besides, approximations (Son & Baral 2001) provide an alternative. They are both briefly reviewed in the next subsections, suppose that an action theory $(\mathcal{D}, \Delta)$ is given.

Possible World Semantics

The possible world semantics is defined based on the transition function Φ^c in Eq. (3). A state s containing a partial state δ is called a *completion* of δ. By $ext(\delta)$ we denote the set of all completions of δ. Observe that the intersection of all the states in $ext(\delta)$ is δ. For a set Δ of partial states, let $ext(\Delta) = \cup_{\delta \in \Delta} ext(\delta)$. An action theory $(\mathcal{D}, \Delta)$ is said to entail a query (4) with respect to the possible world semantics, denoted by,

$$(\mathcal{D}, \Delta) \models_P \varphi \textbf{ after } \alpha$$

if for every $s \in ext(\Delta)$, $\Phi^c(\alpha, s) \neq \perp$ and φ is true in $\Phi^c(\alpha, s)$.

Example 3 Consider the domain $\mathcal{D}_1$ in Example 1 and let $\Delta_1 = \{\emptyset\}$. We have $ext(\Delta_1) = \{s_0, s_1, s_2, s_3\}$ where $s_0 = \{armed, clogged\}$, $s_1 = \{armed, \neg clogged\}$, $s_2 = \{\neg armed, clogged\}$, and $s_3 = \{\neg armed, \neg clogged\}$.

For state s_0 and action $flush$, we have $e(flush, s_0)$=$\{armed, \neg clogged\}$. Hence,

$$Res^c(flush, s_0) = \{armed, \neg clogged\} = s_1$$

Likewise, we have

$$Res^c(flush, s_1) = \{armed, \neg clogged\} = s_1$$
$$Res^c(flush, s_2) = \{\neg armed, \neg clogged\} = s_3$$
$$Res^c(flush, s_3) = \{\neg armed, \neg clogged\} = s_3$$

On the other hand, we can check that

$$Res^c(dunk, s_1) = Res^c(dunk, s_3) = s_2$$

So, for every $s \in ext(\Delta_1)$ we have

$$\Phi^c([flush; dunk], s) = s_2$$

This implies that

$$(\mathcal{D}_1, \Delta_1) \models_P \neg armed \textbf{ after } [flush; dunk]$$

For the domain $\mathcal{D}_2$ in Example 2, let $\Delta_2 = \{\emptyset\}$. Then, the four possible initial states are $ext(\Delta_2) = \{\{g, h\}, \{g, \neg h\}, \{\neg g, h\}, \{\neg g, \neg h\}\}$.

We have

$$\Phi^c([a], \{g, h\}) = \Phi^c([a], \{\neg g, h\}) = \{g, h\}$$
$$\Phi^c([a], \{g, \neg h\}) = \Phi^c([a], \{\neg g, \neg h\}) = \{g, \neg h\}$$

Hence, g holds in the final state after a is executed. That is,

$$(\mathcal{D}_2, \Delta_2) \models_P g \textbf{ after } [a]$$

□

0-Approximation

The 0-approximation is introduced in (Son & Baral 2001). Instead of using the transition function between states (Res^c) to compute the result of the execution of an action, it defines another transition function, denoted by Res, between partial states.

For a partial state δ and an action a executable in δ, let

- $e(a, \delta) = \{l \mid$ there exists a **causes** l **if** ψ in $\mathcal{D}$ s.t. ψ hold in $\delta\}$, and
- $pe(a, \delta) = \{l \mid$ there exists a **causes** l **if** ψ in $\mathcal{D}$ s.t. ψ possibly holds in $\delta\}$.

Intuitively, $e(a, \delta)$, $pe(a, \delta)$ are sets of literals that *must hold* and *may hold*, respectively, after executing a in δ. Observe that the definition of $e(a, \delta)$ extends the definition of $e(a, s)$ described in the previous section to the case of partial states. The transition function Res is defined by:

$$Res(a, \delta) = \begin{cases} (\delta \cup e(a, \delta)) \setminus \neg pe(a, \delta) & \\ & \text{if } a \text{ is executable in } \delta \\ \perp & \text{otherwise} \end{cases}$$

Similarly to Res^c, the function Res is extended to define the partial state after executing a sequence of actions in a given partial state. The new function is called Φ and defined similarly to Φ^c in Eq. (3). The 0-entailment, denoted by $\models_A$, is defined as follows (recall that Δ is a set partial states). $(\mathcal{D}, \Delta)$ entails a query φ **after** α w.r.t. the 0-approximation, denoted by

$$(\mathcal{D}, \Delta) \models_A \varphi \textbf{ after } \alpha,$$

if for every $\delta \in \Delta$, $\Phi(\alpha, \delta) \neq \perp$ and φ is true in $\Phi(\alpha, \delta)$.

Example 4 Consider the action theory $(\mathcal{D}_1, \Delta_1)$ in Example 3. We have

$$e(flush, \emptyset)=\{\neg clogged\} \text{ and } pe(flush, \emptyset)=\{\neg clogged\}$$

Hence, $\delta_1 = Res(flush, \emptyset) = \{\neg clogged\}$.

Furthermore, we have

$$e(dunk, \delta_1)=\{clogged\} \text{ and}$$
$$pe(dunk, \delta_1)=\{clogged, \neg armed\}$$

Thus, $Res(dunk, \delta_1) = \{clogged\}$. Accordingly we have

$$\Phi([flush; dunk], \emptyset) = \{clogged\}$$

This implies that

$$(\mathcal{D}_1, \Delta_1) \not\models_A \neg armed \textbf{ after } [flush; dunk]$$

For the action theory $(\mathcal{D}_2, \Delta_2)$, the only initial partial state is $\delta = \emptyset$. However, because neither h nor $\neg h$ holds in this partial state, the action a is not executable in δ. As a result, we have

$$\Phi([a], \delta) = \perp$$

Hence,

$$(\mathcal{D}_2, \Delta_2) \not\models_A g \textbf{ after } [a]$$

□

A Sufficient Condition for the Completeness of $\models_A$

As discussed previously, the main disadvantage of the 0-approximation is its incompleteness if Δ — the set of initial partial states — does not contain sufficient information for its reasoning. For instance, Examples 3 and 4 show that

$$(\mathcal{D}_1, \Delta_1) \not\models_A \neg armed \textbf{ after } [flush; dunk]$$

while

$$(\mathcal{D}_1, \Delta_1) \models_P \neg armed \textbf{ after } [flush; dunk]$$

In the introductory example, we show that it is possible to make $\models_A$ complete with respect to $\models_P$ without having to examine all possible initial states, by partitioning Δ_1 into the set of partial states $\Delta_1^* = \{\{armed\}, \{\neg armed\}\}$.

Why do we chose the set of fluents $\{armed\}$ but not $\{clogged\}$ to partition Δ_1, although both $armed$ and $clogged$ are unknown in the initial state? In other words, why considering the truth values of $armed$ separately in the beginning *influences* the outcomes of the reasoning process whereas this is not true for $clogged$? In this section, we will provide an answer to this question.

Let us formalize the problem. First, we define what it means by "$\models_A$ is complete with respect to $\models_P$".

Definition 1 *An action theory $(\mathcal{D}, \Delta^*)$ (under the 0-approximation) is said to be* complete *with $(\mathcal{D}, \Delta)$ (under the possible world semantics) if for every formula φ and action sequence α, we have that*

$(\mathcal{D}, \Delta^*) \models_A \varphi$ **after** α if and only if $(\mathcal{D}, \Delta) \models_P \varphi$ **after** α.

Then the problem of our interest is: *Given an action theory $(\mathcal{D}, \Delta)$, find a set Δ^* of partial states such that $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \Delta)$.*

We will solve this problem by answering step by step the following questions:

Question 1: What is a sufficient condition for $(\mathcal{D}, \{\delta\})$ to be complete (with itself)?

Question 2: How can we modify $(\mathcal{D}, \{\delta\})$ into $(\mathcal{D}, \Delta^*)$ such that $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \{\delta\})$?

Question 3: How can we modify $(\mathcal{D}, \Delta)$ into $(\mathcal{D}, \Delta^*)$ such that $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \Delta)$?

In answering Question 1 we rely on our earlier observation: there are certain fluents whose values need to be known in δ if $(\mathcal{D}, \{\delta\})$ were to be complete. In other words, the values of other fluents depend on the values of some fluents. To make it precise, we introduce the notion of dependencies between fluents and between actions and fluents as follows.

Definition 2 *A literal l_1* depends on *a literal l_2, written $l_1 \lhd l_2$, if*

- $l_1 \equiv l_2$,
- *there exists a dynamic law*
  $$a \textbf{ causes } l_1 \textbf{ if } \psi$$
  in $\mathcal{D}$ such that $l_2 \in \psi$,
- *there exists l_3 such that $l_1 \lhd l_3$ and $l_3 \lhd l_2$, or*
- $\neg l_1 \lhd \neg l_2$.

An action a depends on *a literal l, written $a \lhd l$, if*

- *there exists an executability condition*
  $$\textbf{executable } a \textbf{ if } \psi$$
  in $\mathcal{D}$ such that $l \in \psi$, or
- *there exists l_1 such that $a \lhd l_1$ and $l_1 \lhd l$.*

For each literal l (resp. action a), we denote by $\Omega(l)$ (resp. $\Omega(a)$) the set of literals that l (resp. a) depends on. As an example, for the domain $\mathcal{D}_1$, we have

$\Omega(clogged) = \{clogged\}$
$\Omega(\neg clogged) = \{\neg clogged\}$
$\Omega(armed) = \{armed, \neg armed\}$
$\Omega(\neg armed) = \{armed, \neg armed\}$
$\Omega(dunk) = \{\neg clogged\}$ $\quad$ $\Omega(flush) = \emptyset$

and for the domain $\mathcal{D}_2$, we have

$\Omega(g) = \{g\}$ $\quad$ $\Omega(h) = \{h\}$
$\Omega(\neg g) = \{\neg g\}$ $\quad$ $\Omega(\neg h) = \{\neg h\}$ $\quad$ $\Omega(a) = \{h, \neg h\}$

Intuitively, $l_1 \lhd l_2$ means that knowledge about l_2 might be needed in reasoning about the truth value of l_1 after the execution of some sequence of actions; $a \lhd l$ means that l might have influence on determining the executability of action a. In the next definition, we characterize a set of states S for which the 0-approximation— starting from the partial state $\delta = \cap_{s \in S} s$— is complete.

Definition 3 *Let S be set of states and δ be the intersection of all states in S. We say that S is* approximatable *if*

1. *there exists no literal l such that for every $s \in S$, $l \lhd l_1$ for some $l_1 \in s \setminus \delta$, and*
2. *there exists no action a such that for every $s \in S$, $a \lhd l_1$ for some $l_1 \in s \setminus \delta$.*

Example 5 Consider the domain $\mathcal{D}_1$. Let $\delta = \emptyset$. Then, $S_1 = ext(\delta)$ is not approximatable because (i) for every $s \in S_1$, either $armed$ or $\neg armed$ belongs to $s \setminus \delta$ (Recall that $\delta = \cap_{s \in ext(\delta)} s$); and (ii) $l = armed$ depends on both $armed$ and $\neg armed$. For the same reason, the sets $ext(\{clogged\})$ and $ext(\{\neg clogged\})$ are not approximatable.

On the other hand, the sets $ext(\{armed\})$ and $ext(\{\neg armed\})$ are approximatable because there exists no literal l (or action a) which depends on both $clogged$ and $\neg clogged$.

For the domain $\mathcal{D}_2$, the set of states $S_2 = ext(\emptyset)$ is not approximatable because for every $s \in S_2$, either h or $\neg h$ belongs to $s \setminus \emptyset$ and a depends on both h and $\neg h$. We can easily check that the sets $ext(\{g\})$ and $ext(\{\neg g\})$ are not approximatable either. $ext(\{h\})$ and $ext(\{\neg h\})$, however, are approximatable sets. □

The following proposition shows an interesting property of an approximatable set.

Proposition 1 *Let S be an approximatable set of states, δ be the intersection of states in S, and a be an action executable in S. Then, a is also executable in δ and furthermore*

$$Res(a, \delta) = \bigcap_{s' \in Res^c(a,S)} s'$$

Sketch of Proof. We first show that a is executable in δ. Suppose otherwise. Because a is executable in S, for every $s \in S$, $\mathcal{D}$ contains an executability condition (2) such that $\psi \subseteq s$; furthermore, by our assumption, $\psi \not\subseteq \delta$. This means that there exists a literal $l \in s \setminus \delta$ such that a depends on l. This violates the condition that S is approximatable.

Next, we show that $\bigcap_{s' \in Res^c(a,S)} s' \subseteq Res(a, \delta)$. Suppose otherwise, that is, there exists l such that $l \in \bigcap_{s' \in Res^c(a,S)} s'$ but $l \notin Res(a, \delta)$. We can prove that for every $s \in S$, there exists $l_1 \in s \setminus \delta$ such that $l \lhd l_1$. This violates the condition that S is approximatable.

The above results together with the soundness of the 0-approximation allow us to conclude the proposition. □

The intuitive meaning of Proposition 1 is that if $ext(\delta)$ is approximatable then the theory $(\mathcal{D}, \{\delta\})$ is "complete" after performing *a single action*. It, however, does not imply that $(\mathcal{D}, \{\delta\})$ is complete after performing *any sequence of actions* because after an action is performed, one may think that we could loose the approximatability property of the set of possible states. Nevertheless, the following proposition shows that this property is preserved along the course of action execution.

Proposition 2 *For every action a executable in S, if S is approximatable then so is $Res^c(a, S)$.*

Sketch of Proof. Let δ and δ' be the intersections of all states in S and $Res^c(a, S)$ respectively. Consider a state $s \in S$. Let s' be the successor state of s after a. Then, we can prove the following result:

$$\forall(l_1 \in s' \setminus \delta')\exists(l_2 \in s \setminus \delta).l_1 \lhd l_2$$

Then if $Res^c(a, S)$ is not approximatable then S is also not approximatable and thus, this cannot happen. □

From Propositions 1 and 2, we have the following theorem.

Theorem 1 *An action theory $(\mathcal{D}, \{\delta\})$ is complete if $ext(\delta)$ is approximatable.*

Sketch of Proof. Let α be a sequence of actions. From Propositions 1 and 2, we have that

$$\Phi(\alpha, \delta) = \bigcap_{s \in ext(\delta)} \Phi^c(\alpha, \delta)$$

By Definition 1 and the definitions of $\models_A$ and $\models_P$, this means that $(\mathcal{D}, \{\delta\})$ is complete with itself. □

This theorem serves as a sufficient condition for $(\mathcal{D}, \{\delta\})$ to be complete and provides an answer to Question 1. As can be seen in Example 5, for the domain $\mathcal{D}_1$, the sets $ext(\{armed\})$ and $ext(\{\neg armed\})$ are approximatable sets. Thus, the above theorem implies that the action theories $(\mathcal{D}_1, \{\{armed\}\})$ and $(\mathcal{D}_1, \{\{\neg armed\}\})$ are complete.

Observe that the approximatability of $ext(\delta)$ is a sufficient but not necessary condition for the completeness of $(\mathcal{D}, \delta)$. For example, it is easy to check that the theory $(\{a$ **causes** f **if** g, a **causes** f **if** $g, \neg f\}, \{\{g\}\})$ is complete. However, $ext(\{g\})$ is not an approximatable set of states because it violates the first condition of Definition 3.

Theorem 1 suggests a way to address Question 2, i.e., we can partition the set of possible initial states $ext(\delta)$ into subsets such that each of them is approximatable. This can be done by determining a decisive set of fluents for $(\mathcal{D}, \{\delta\})$ which is defined as follows.

Definition 4 *A set F of fluents is called a* decisive *set for $(\mathcal{D}, \{\delta\})$, where δ is a partial state, if the following conditions are satisfied*

- *every fluent $f \in F$ is unknown in δ, and*
- *for every interpretation I of F[6], $ext(\delta \cup I)$ is an approximatable set.*

[6]An interpretation of F is a consistent set of literals σ such that there exists a set $G \subseteq F$ and $\sigma = \{f \mid f \in G\} \cup \{\neg f \mid f \in F \setminus G\}$.

By this definition, $F_1 = \{armed\}$ and $F_2 = \{armed, clogged\}$ are decisive sets for $(\mathcal{D}_1, \Delta_1)$ whereas $F_3 = \{clogged\}$ is not. The following theorem shows an important property of a decisive set.

Theorem 2 *Let $(\mathcal{D}, \{\delta\})$ be an action theory and let F be a decisive set for $(\mathcal{D}, \{\delta\})$. Define*

$$\Delta^* = \{\delta \cup I \mid I \text{ is an interpretation of } F\}$$

Then, $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \{\delta\})$.

Sketch of Proof. Let δ^* be a partial state in Δ^*. By the definition of F, $ext(\delta^*)$ is approximatable. From this and by Theorem 1, for any sequence α of actions, we have

$$\Phi(\alpha, \delta^*) = \bigcap_{s \in ext(\delta^*)} \Phi^c(\alpha, s)$$

On the other hand, notice that

$$ext(\delta) = \bigcup_{\delta^* \in \Delta^*} ext(\delta^*)$$

Accordingly, we can conclude that $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \{\delta\})$. □

This theorem implies that Question 2 can be answered if a decisive set for $(\mathcal{D}, \{\delta\})$ can be found. Trivially, for every δ, the set U_δ of all unknown fluents in δ is *always* a decisive set for $(\mathcal{D}, \{\delta\})$. It is, however, important to note that the number of interpretations of U_δ is exponential in the size of U_δ. Hence, given a δ, we wish to find a decisive set for $(\mathcal{D}, \{\delta\})$ that is as small as possible (w.r.t. set inclusion $\subseteq$). To do so, we develop an algorithm for computing a decisive set (Figure 2). The algorithm is based on the concepts of dependencies in Definition 2.

```
DECISIVE(D, δ)
INPUT: a domain description D a partial state δ
OUTPUT: a decisive set of fluents for (D, δ)
BEGIN
  F = ∅
  compute dependencies between literals
  compute dependencies between actions and literals
  for each fluent f unknown in δ do
      if there exists l s.t. l depends on both f and ¬f or
      an action a s.t. a depends on both f and ¬f
      then F = F ∪ {f}
  return F;
END
```

Figure 2: Computing a decisive set of fluents for $(\mathcal{D}, \{\delta\})$

The following proposition shows that the algorithm correctly computes a decisive set.

Proposition 3 *The set of fluents returned by* DECISIVE($\mathcal{D}$, δ) *is a decisive set for $(\mathcal{D}, \{\delta\})$.*

Sketch of Proof. Let F be the set of fluents returned by the algorithm. First, notice that F contains only fluent unknown in δ. Second, we can prove that for every interpretation I of F, $ext(\delta^*)$ is an approximatable set, where $\delta^* = \delta \cup I$. Therefore, by Definition 4, F is a decisive set for $(\mathcal{D}, \{\delta\})$. □

Example 6 Consider the action theory $(\mathcal{D}_1, \Delta_1)$. Then, the decisive set returned by the algorithm for $(\mathcal{D}_1, \Delta_1)$ is $\{armed\}$. Let Δ_1^* be the partition of Δ_1 over $\{armed\}$, that is, $\Delta_1^* = \{\{armed\}, \{\neg armed\}\}$. Hence, by Theorem 2, $(\mathcal{D}_1, \Delta_1^*)$ is complete with $(\mathcal{D}_1, \Delta_1)$.

For the action theory $(\mathcal{D}_2, \Delta_2)$, the returned decisive set is $\{h\}$. The partition of Δ_2 over $\{h\}$ is $\Delta_2^* = \{\{h\}, \{\neg h\}\}$. Then, by Theorem 2, $(\mathcal{D}_2, \Delta_2^*)$ is complete with $(\mathcal{D}_2, \Delta_2)$. □

Although simple and somewhat naive, the algorithm is worth some discussion. According to the algorithm, an unknown fluent f belongs to the returned set F if there exists a literal l or an action a that depends on both f and $\neg f$. As can be seen in the proof of Proposition 3, this guarantees that F is a decisive set because for every interpretation I of F, $ext(\delta \cup I)$ is approximatable. The main weakness of this algorithm is that it *does not* guarantee the minimality of F. Observe that an implementation based on the definition of an approximatable set (Definition 3) might return a smaller decisive set. Nevertheless, we adopt the above algorithm in the development of our planner (with a little change, to be described in the next section) for two reasons. First, it is computationally efficient (its run time is polynomial in the size of the domain). Second, for a majority of the benchmark problems, we notice that the decisive set returned by the algorithm is empty set which is already as small as possible.

Once a decisive set F_δ for $(\mathcal{D}, \{\delta\})$ can be computed (i.e., Question 2 is answered), we can easily find a solution to Question 3 as shown in the following theorem.

Theorem 3 *Let $(\mathcal{D}, \Delta)$ be an action theory. For every $\delta \in \Delta$, let F_δ be a decisive set for $(\mathcal{D}, \{\delta\})$. Define*

$$\Delta^* = \bigcup_{\delta \in \Delta} \{\delta \cup I \mid I \text{ is an interpretation of } F_\delta\}$$

Then, $(\mathcal{D}, \Delta^*)$ is complete with $(\mathcal{D}, \Delta)$.

Sketch of Proof. Let α be a sequence of actions and φ be an arbitrary formula.

Observe that $(\mathcal{D}, \Delta) \models_P \varphi$ **after** α iff for every $\delta \in \Delta$, $(\mathcal{D}, \{\delta\}) \models_P \varphi$ **after** α; and $(\mathcal{D}, \Delta^*) \models_A \varphi$ **after** α iff for every $\delta^* \in \Delta^*$, $(\mathcal{D}, \{\delta^*\}) \models_P \varphi$ **after** α. Furthermore, by the definition of decisive sets (Definition 4) and by the definition of Δ^*, we have that for each $\delta \in \Delta$ there exists $\delta^* \in \Delta^*$ such that $(\mathcal{D}, \{\delta\}) \models_P \varphi$ **after** α iff $(\mathcal{D}, \{\delta^*\}) \models_A \varphi$ **after** α and vice versa.

The above observations imply that for any formula φ and action sequence α

$$(\mathcal{D}, \Delta) \models_P \varphi \textbf{ after } \alpha \text{ iff } (\mathcal{D}, \Delta^*) \models_A \varphi \textbf{ after } \alpha$$

□

Application to Conformant Planning

In this section, we will present a sound and complete conformant planner that is based on the result in the previous section. We first review the conformant planning problem and then discuss how such a conformant planner can be built.

A *conformant planning problem* (or planning problem for short) $\mathcal{P}$ is a tuple $\langle \mathcal{D}, \Delta, \delta_f \rangle$ where $(\mathcal{D}, \Delta)$ is an action theory and δ_f is a partial state representing the goal. A *solution* to $\mathcal{P}$ is an action sequence α such that $(\mathcal{D}, \Delta) \models_P \delta^f$ **after** α. For instance, $\mathcal{P}_1 = \langle \mathcal{D}_1, \{\emptyset\}, \{\neg clogged\} \rangle$ and $\mathcal{P}_2 = \langle \mathcal{D}_1, \{\emptyset\}, \{\neg armed\} \rangle$ are planning problems and $\alpha_1 = [flush]$ and $\alpha_2 = [flush; dunk]$ are their solution respectively.

In our previous work (Son *et al.* 2005b; 2005a), we used the 0-approximation in the development of a suite of conformant planners, named CPA, for domains with state constraints. The main weakness is that these planners are incomplete. As an example, for the problem $\mathcal{P}_2$, CPA returns no solution. If we wish to make it return a solution, we would have to *manually* encode Δ as $\{\{armed\}, \{\neg armed\}\}$. It follows from Theorem 3 that we can indeed *automatically* add to Δ the necessary information to make CPA complete. We will now discuss this idea in more details.

A straightforward way to achieve the completeness of CPA is as follows. For each $\delta \in \Delta$, (1) compute a decisive set F_δ for $(\mathcal{D}, \{\delta\})$ based on the algorithm DECISIVE($\mathcal{D}$,δ) (Figure 2); (2) then generate Δ^* from Δ and the decisive sets F_δ; and, finally, (3) use $(\mathcal{D}, \Delta^*, \delta_f)$ instead of $(\mathcal{D}, \Delta, \delta_f)$ as input to CPA. Although this method guarantees that CPA is complete, it does not take into consideration the information about the goal. For this reason, instead of using the algorithm DECISIVE($\mathcal{D}$,δ) in the second step, we use a modified version called DECISIVE($\mathcal{D}$,δ,δ_f) (Figure 3). This algorithm accepts a third parameter, δ^f, which represents the goal, and generates a set of decisive fluents which provides the planner enough information to solve the planning problem. The modification is fairly simple: in the body of the algorithm DECISIVE($\mathcal{D}$,δ), we replace "... exists l s.t. ..." with "... exists $l \in \delta^f$ s.t. ...".

DECISIVE($\mathcal{D}$, δ, δ^f)
INPUT: a domain description $\mathcal{D}$, partial states δ and δ^f
OUTPUT: a decisive set of fluents for $\langle \mathcal{D}, \{\delta\}, \delta^f \rangle$
BEGIN
  $F = \emptyset$
  compute dependencies between literals
  compute dependencies between actions and literals
  for each fluent f unknown in δ **do**
    if there exists $l \in \delta^f$ s.t. l depends on both f and $\neg f$ or an action a s.t. a depends on both f and $\neg f$
    then $F = F \cup \{f\}$
  return F;
END

Figure 3: DECISIVE($\mathcal{D}$, δ, δ^f) - Computing a decisive set of fluents for $\langle \mathcal{D}, \{\delta\}, \delta^f \rangle$

It is easy to see that the following theorem holds.

Theorem 4 *Let $(\mathcal{D}, \Delta, \delta_f)$ be a planning problem. For every $\delta \in \Delta$, let F_δ be* DECISIVE($\mathcal{D}$, δ, δ^f). *Then, α is a solution to $\mathcal{P}$ iff $(\mathcal{D}, \Delta^*) \models_A \delta^f$ **after** α where $\Delta^* = \bigcup_{\delta \in \Delta} \{\delta \cup I \mid I \text{ is an interpretation of } F_\delta\}$.*

The correctness of Theorem 4 shows that building a complete conformant planner based on the 0-approximation is feasible. We therefore develop a conformant planner, called CPA$^+$. The implementation of CPA$^+$ is based on the source code of CPA (Son *et al.* 2005b), adding a module for computing decisive sets for partial states $\delta \in \Delta$ based on the algorithm presented in Figure 3 and generating Δ^* from Δ. In addition, everything related to static causal laws is removed. Inherited from CPA, CPA$^+$ is a forward, best-first search planner with the number of fulfilled subgoals as its heuristic function.

We compare CPA$^+$ with three planners Conformant-FF (CFF) (Brafman & Hoffmann 2004), KACMBP(Cimatti, Roveri, & Bertoli 2004), and POND (Cushing & Bryce 2005) because to the best of our knowledge they belong to the fastest conformant planners in most of the benchmark domains in the literature. The domains used in our experiments are the bomb-in-the-toilet (bomb), ring, logistics, and cleaner. In the bomb domain, we experimented with $p = 10, 20, 50, 100$ packages and $t = 1, 5, 10$ toilets. In the logistics domain we did experiments with 5 problems, corresponding to $l = 2, 3, 4$ and $c = p = 2, 3$, where l, c, and p are the numbers of locations per city, cities, and packages respectively, (only logistics(4,2,2) is not available). In the ring domain, we tested with n =2,5,10, and 20, where n is the number of rooms. In the cleaner domain (Son *et al.* 2005b), we tested with 6 problems corresponding to $n = 2, 5$ and $p = 10, 50, 100$ respectively, where n is the number of rooms and p is the number of objects.

All experiments were run on a 2.4 GHz CPU, 768MB RAM machine, running Slackware 10.0 operating system. Time limit is set to half an hour. The testing results are shown in Tables 1–4. In each table, columns 1–3 show the characteristics of the problem: the number of initial partial states (i.e., size of Δ), the total number of fluents, and the number of unknown fluents in each initial partial state. The next columns report the length of the returned solution and the running time of the planner. Times are shown in seconds; 'TO", "AB", and "NA" indicate that the corresponding planner ran out of time without returning a solution, that the planner stopped abnormally, and that the problem is not applicable, respectively. We ran two versions of the planner, one of which uses the possible world semantics (CPA*) and the other uses the 0-approximation semantics embedded with the module of computing decisive sets (CPA$^+$).

As can be seen in Table 1, CFF is superior to KACMBP and CPA$^+$ over the logistics domain. It took only 0.14 seconds to solve the hardest instance $logistics(4, 3, 3)$ while both KACMBP and CPA$^+$ reported a time out. CPA$^+$ is better than KACMBP over the first three instance but slower on $logistics(3, 3, 3)$. It should be noted here that one of the characteristics of this domain is that all the partial states in Δ are complete, i.e., they are states indeed; hence, using the 0-approximation to implement CPA$^+$ does not help solve the problems more quickly than if it is implemented using possible world semantics (CPA*). Another factor that may cause the slow performance of CPA$^+$ is because of its simple heuristic function.

In the ring domain (Table 2), KACMP is the best. But this domain is really problematic for CFF. As explained in (Brafman & Hoffmann 2004), it is because of the lack of informativity of the heuristic function in the presence of nonunary effect conditions and the problem with checking repeated states. Both POND and CFF can solve only the first problem within the time limit. CPA$^+$ is much better than CFF and POND but slower than KACMBP. In this domain, and in all the other domains that follow as well, we can see there is a big difference between the performance of CPA* and CPA$^+$. The reason for the good performance of CPA$^+$ over CPA* is because the uncertainty degree of the problems in these domains is high, thus, making the performance of CPA* gets worse quickly because it has to consider all possible initial states which is exponential in the number of unknown fluents.

In the bomb domain (Table 3), CPA$^+$ outperforms all the other planners. It took only around 7 seconds to solve the hardest problem, bomb(100,10), whereas that solving time for KACMP and CFF are more than 35 seconds and 100 seconds respectively; POND reported a time out. The reason for this good result of CPA$^+$ is because this domain exposes a very high degree of uncertainty (i.e., the number of unknown fluents in the initial partial state is almost the same as the total number of fluents). In addition, the planner detects that none of the unknown fluents is needed for its reasoning. As a result, at any time during the search, the planner needs to consider only one partial state and the size of search space is rather small. Similarly to the bomb domain, CPA$^+$ also works well with the cleaner domain (Table 4). For the hardest problem, it took around 68 seconds, outputting a plan of length 504. None of the other planners was able to solve the last problem within the time limit[7]. Among the others, CFF is the best. It outperforms both KACMBP and POND on this domain. The reason for good performance of CPA$^+$ on this domain can be explained similarly as with the bomb domain.

Finally, we would like to mention that, in this paper, we concentrate on comparing our planner CPA$^+$ with other conformant planners with the same capability. Among other things, the representation language employed in the discussed planners is, in one way or another, a propositional language with limited expressive power. For this reason, we do not compare CPA$^+$ with the PKS system (Petrick & Bacchus 2002) (improved in (Petrick & Bacchus 2004)) which employs a richer representation language and the knowledge-based approach to reason about effects of actions in the presence of incomplete information. In a near future, we plan to investigate the relationship between CPA$^+$ and PKS and other conditional planners.

Discussion and Related Work

One of the main advantages of the 0-approximation is its lower complexity in comparison with the possible world semantics in planning and reasoning when the initial state (Δ)

[7]CFF stopped because the maximum length of a plan is exceeded. We believe that it can be easily fixed by increasing this constant in the source code

Problem	$\|\Delta\|$	$\|\mathbf{F}\|$	Unkwn.	KACMBP	POND	CFF	CPA*	CPA$^+$
logistics(2,2,2)	4	20	0	14/ 0.19	/ NA	16 / 0.03	9 / 0.047	9 / 0.058
logistics(2,3,3)	8	39	0	34/ 355.96	/ NA	24 / 0.06	48 / 2.24	48 / 2.217
logistics(3,2,2)	9	26	0	17/ 2.1	/ NA	20 / 0.06	44 / 1.384	44 / 1.363
logistics(3,3,3)	27	51	0	40/ 29.8	/ NA	34 / 0.12	350 / 93.905	350 / 93.173
logistics(4,3,3)	64	63	0	/ TO	/ NA	37 / 0.14	/ TO	/ TO

Table 1: Logistics Domain

Problem	$\|\Delta\|$	$\|\mathbf{F}\|$	Unkwn.	KACMBP	POND	CFF	CPA*	CPA$^+$
ring(2)	2	6	4	5/ 0.00	6 / 0.156	7 / 0.06	5 / 0.009	5 / 0.002
ring(3)	3	9	6	8/ 0.00	8 / 0.089	15 / 0.23	8 / 0.094	8 / 0.004
ring(4)	4	12	8	11/ 0.00	13 / 0.251	26 / 3.86	11 / 0.773	11 / 0.009
ring(5)	5	15	10	14/ 0.00	17 / 0.967	45 / 63.67	15 / 5.501	15 / 0.018
ring(10)	10	30	20	29/ 0.02	/ TO	/ TO	/ TO	30 / 0.111
ring(15)	15	45	30	44/ 0.04	/ TO	/ TO	/ TO	45 / 0.38
ring(20)	20	60	40	59/ 0.15	/ TO	/ TO	/ TO	60 / 0.928
ring(25)	25	75	50	74/ 0.32	/ TO	/ TO	/ TO	75 / 1.921

Table 2: Ring Domain

consists of a single partial state[8] (Baral, Kreinovich, & Trejo 2000). The price one has to pay for this lower complexity is the incompleteness of reasoning and planning tasks. It is therefore natural to see this advantage (low complexity) to disappear even when we deal with action theories with more than one initial partial state (i.e., $\mid \Delta \mid > 1$).

As can be seen, complete reasoning using 0-approximation and decisive sets of fluents offers certain advantages over reasoning based on possible world semantics if (partial) states are represented explicitly. Given a domain $\mathcal{D}$ with n fluents and an initial partial state δ with a decisive set F_δ, the number of partial states that need to be considered by the 0-approximation is $2^{|F_\delta|}$ whereas the number of states which need to be considered by the possible world semantics is $2^{n-|\delta|}$. Because $|F_\delta| \leq n - |\delta|$, reasoning based on the 0-approximation will certainly be more efficient than reasoning based on the possible world semantics. It is interesting to note that in most of the benchmarks for conformant planning, we found that $F_\delta = \emptyset$ or $|F_\delta|$ is much smaller than $n - |\delta|$. This explains why CPA$^+$, even built with a simple heuristic, can achieve very good performance as shown in the previous section. Nevertheless, due to the fact that it represents partial states explicitly, CPA$^+$ will not be able to work with domains for which F_δ is large for some initial partial state δ because enumerating all the possible initial partial states might already take exponential time in the size of F_δ. We consider this as a disadvantage of CPA$^+$and are investigating different representation framework to address this issue.

We believe that our work can be applied not only to reasoners that represent (partial) states explicitly but also to those that represent (partial) states implicitly. We did some experiments with CFF, KACMBP, and POND by adding to the bomb domain several irrelevant fluents and specifying them to be unknown in the initial state. We observed that for the bomb domain, when the number of fluents added is the same as the number of fluents of the original problem, these planners ran on the modified problem around 2, 10 and 2 times, respectively, i.e., slower than on the original problem. This suggests that if these planners can remove the unnecessary fluents from consideration, they would yield better performance.

In the past, a number of researchers, e.g., (Haslum & Jonsson 2000; Lifschitz & Ren 2004; Nebel, Dimopoulos, & Koehler 1997), already realized that a planning problem may contain much irrelevant information, including irrelevant fluents, irrelevant actions, etc. In most of these work, if irrelevant information is found then it can be safely removed from the problem. In our approach, on the contrary, if a fluent does not belong to a decisive set, it does not mean that we can safely remove it from the theory without affecting the reasoning process. Rather, it only means that considering its truth value separately in the beginning is not necessary. For example, in the bomb domain, we cannot remove $clogged$ although it does not belong to the decisive set $\{armed\}$. In the following, we briefly relate our work to others' work.

In (Lifschitz & Ren 2004), the idea of relevant actions of a planning problem $\mathcal{P}$ with the action domain $\mathcal{D}$ is made precise by the notion of an isolated set. Basically, an isolated set σ with respect to $\mathcal{D}$ could be viewed as a partition of the original domain. If every fluent appearing in the goal is contained in an isolated set σ, then solutions to $\mathcal{P}$ can be found by using σ instead of $\mathcal{D}$. An isolated set differs from a decisive set of fluents in that *(i)* it contains not only fluents but also actions and laws; *(ii)* it does not take into consideration the knowledge about the initial state. We observe that a problem might not have a non-trivial isolated set (the complete domain is the trivial isolated set) but can still have a decisive set of fluents. For example, for the planning problem $\mathcal{P}_3 = \langle \mathcal{D}_1, \{\emptyset\}, \{\neg armed, clogged\}\rangle$, no non-trivial

[8] We observe that most of the benchmarks in conformant planning satisfy this property.

Problem	$\|\Delta\|$	$\|\mathbf{F}\|$	Unkwn.	KACMBP	POND	CFF	CPA*	CPA$^+$
bomb(5,1)	1	6	5	9/ 0	9 / 0.038	9 / 0.03	9 / 0.012	9 / 0.003
bomb(10,1)	1	11	10	19/ 0.01	19 / 0.078	19 / 0.05	19 / 1.038	19 / 0.007
bomb(20,1)	1	21	20	39/ 0.05	39 / 0.578	39 / 0.17	/ TO	39 / 0.036
bomb(50,1)	1	51	50	99/ 0.51	99 / 28.695	99 / 5.33	/ TO	99 / 0.312
bomb(100,1)	1	101	100	199/ 3.89	199 / 682.33	199 / 121.8	/ TO	199 / 2.284
bomb(5,5)	1	10	5	5/ 0.04	5 / 0.1	5 / 0.04	5 / 0.147	5 / 0.006
bomb(10,5)	1	15	10	15/ 0.09	15 / 0.654	15 / 0.07	15 / 6.006	15 / 0.02
bomb(20,5)	1	25	20	35/ 0.3	35 / 7.284	35 / 0.16	/ TO	35 / 0.076
bomb(50,5)	1	55	50	95/ 1.66	95 / 348.28	95 / 4.7	/ TO	95 / 0.689
bomb(100,5)	1	105	100	195/ 6.92	/ TO	195 / 113.95	/ TO	195 / 4.507
bomb(5,10)	1	15	5	5/ 0.11	5 / 0.357	5 / 0.03	5 / 0.265	5 / 0.015
bomb(10,10)	1	20	10	10/ 0.3	10 / 2.504	10 / 0.05	10 / 15.003	10 / 0.055
bomb(20,10)	1	30	20	30/ 0.97	30 / 27.69	30 / 0.13	/ TO	30 / 0.154
bomb(50,10)	1	50	50	90/ 5.39	90 / 960.004	90 / 4.04	/ TO	90 / 1.262
bomb(100,10)	1	110	100	190/ 35.83	/ TO	190 / 102.56	/ TO	190 / 7.447

Table 3: Bomb Domain

Problem	$\|\Delta\|$	$\|\mathbf{F}\|$	Unkwn.	KACMBP	POND	CFF	CPA*	CPA$^+$
cleaner(2,5)	1	12	10	11/ 0.01	11 / 0.173	11 / 0.03	11 / 6.878	11 / 0.044
cleaner(2,10)	1	22	20	21/ 0.08	21 / 0.853	21 / 0.07	/ TO	21 / 0.044
cleaner(2,20)	1	42	40	41/ 0.62	41 / 15.875	41 / 0.15	/ TO	41 / 0.092
cleaner(2,50)	1	102	100	101/ 13.55	/ TO	101 / 0.8	/ TO	101 / 1.039
cleaner(2,100)	1	202	200	201/ 185.39	/ TO	201 / 5.72	/ TO	201 / 8.096
cleaner(5,5)	1	30	25	34/ 0.017	29 / 1.469	29 / 0.11	/ TO	29 / 0.02
cleaner(5,10)	1	55	50	56/ 0.096	54 / 12.868	54 / 0.24	/ TO	54 / 0.095
cleaner(5,20)	1	105	100	106/ 7.82	104 / 214.832	104 / 0.85	/ TO	104 / 0.588
cleaner(5,50)	1	255	250	256/ 227.82	/ TO	254 / 14.36	/ TO	254 / 8.73
cleaner(5,100)	1	505	500	/ TO	/ TO	AB /	/ TO	504 / 68.023

Table 4: Cleaner Domain

isolated set can be found but $\{armed\}$ is a decisive set for $(\mathcal{D}, \{\emptyset\})$. We hypothesize that if σ is an isolated set with respect to $\mathcal{D}$ containing the goal δ^f then the set of fluents in σ, which are unknown in an initial state δ, constitutes a decisive set for $(\mathcal{D}, \{\delta\}, \delta^f)$.

In (Haslum & Jonsson 2000), the notion of a reduced operator set (of a planning problem) is defined and algorithms for computing such sets are presented. Intuitively, a reduced operator set consists of operators needed for solving the problem and does not contain any redundant operator (an operator is redundant if it can be replaced by a sequence of operators). They also implemented a preprocessor for computing a reduced operator set of a planning problem and demonstrated its usefulness in various planners. We note that this work concentrates on domains with complete initial state while we focus on domains with incomplete information.

In (Nebel, Dimopoulos, & Koehler 1997), information relevant to a planning problem is selected by backchaining from the goals. This is done as follows. First, given a planning problem, a fact-generation tree – an AND-OR-tree where the AND-nodes are sets of ground facts and the OR-nodes contains a single ground fact; the root is the set of goals – is constructed. Then, based on the structure of this tree, the set of sets of initial facts possibly needed for the goals, called possibility set for the goals, is determined. Based on this set, they propose several methods of selecting "probably relevant" pieces of information. This approach differs from ours in that *(i)* it is intended for planning with complete initial state; and *(ii)* it may exclude information that is indeed relevant, making the problem unsolvable even if it is solvable.

Conclusions and Future Work

In this paper, we develop a method for complete reasoning in the presence of incomplete information based on the 0-approximation. Our idea is based on the notion of a decisive set for a partial state. This decisive set can be used to group the set of possible initial (complete) states into a smaller set of partial states which guarantees the completeness of reasoning based on the 0-approximation w.r.t. reasoning using the possible world approach. We present an algorithm for computing a decisive set of fluents given an action theory. We extend this idea to the conformant planning problem and validate the usefulness of this idea by developing a sound and complete conformant planner called CPA$^+$. The experimental results show that CPA$^+$ is competitive with other state-of-the-art conformant planners, validating the usefulness of the notion of decisive set. In this paper, we concentrate on the definition of a decisive set and its application in conformant planning. Identifying and computing *minimal*

decisive sets are the topics that we plan to further our study in the immediate future. To scale up the planner, we would like to develop a new implementation that does not require the enumeration of the set of initial partial states.

Acknowledgments: We would like to thank Michael Gelfond for the numerous discussions on the topic of the paper which inspired us to carry out this research. The authors are partially supported by NSF grants EIA-0220590 and CNS-0454066.

References

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122:241–267.

Brafman, R., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 355–364. Whistler, Canada: Morgan Kaufmann.

Cimatti, A.; Roveri, M.; and Bertoli, P. 2004. Conformant Planning via Symbolic Model Checking and Heuristic Search. *Artificial Intelligence Journal* 159:127–206.

Cushing, W., and Bryce, D. 2005. State Agnostic Planning Graphs and the application to belief-space planning. In *Proceedings of the the Twentieth National Conference on Artificial Intelligence*.

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

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

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

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

Goldman, R., and Boddy, M. 1994. Representing uncertainty in simple planners. In *KR 94*, 238–245.

Goldman, R., and Boddy, M. 1996. Expressive planning and explicit knowledge. In *AIPS 96*, 110–117.

Haslum, P., and Jonsson, P. 2000. Planning with reduced operator sets. In *AIPS*, 150–158.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proceedings of the 14th Conference on Artificial Intelligence*, 1139–1146. AAAI Press.

Lifschitz, V., and Ren, W. 2004. Irrelevant actions in plan generation (extended abstract). In *IX Ibero-American Workshops on Artificial Intelligence*, 71–78.

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

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In Steel, S., and Alami, R., eds., *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*, volume 1348 of *Lecture Notes in Computer Science*, 338–350. Springer.

Peot, M., and Smith, D. 1992. Conditional non-linear planning. In *First Conference of AI Planning Systems*, 189–197.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, 212–222. AAAI.

Petrick, R. P. A., and Bacchus, F. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the Sixth International Conference on Automated Planning and Scheduling, 2004*, 2–11.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research* 4:287–339.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings of AAAI 98*.

Son, T., and Baral, C. 2001. Formalizing sensing actions - a transition function based approach. *Artificial Intelligence* 125(1-2):19–91.

Son, T. C.; Tu, P. H.; Gelfond, M.; and Morales, R. 2005a. An Approximation of Action Theories of $\mathcal{AL}$ and its Application to Conformant Planning. In *Proceedings of the the 7th International Conference on Logic Programming and NonMonotonic Reasoning*, 172–184.

Son, T. C.; Tu, P. H.; Gelfond, M.; and Morales, R. 2005b. Conformant Planning for Domains with Constraints — A New Approach. In *Proceedings of the the Twentieth National Conference on Artificial Intelligence*, 1211–1216.

Son, T.; Tu, P.; and Baral, C. 2004. Planning with Sensing Actions and Incomplete Information using Logic Programming. In Lifschitz, V., and Niemelä, I., eds., *Proceedings of the 7th International Conference on Logic Programming and NonMonotonic Reasoning Conference (LPNMR'04)*, volume 2923, 261–274. Springer Verlag, LNCS 2923.

Thielscher, M. 2002. Reasoning about actions with CHRs and finite domain constraints. In Stuckey, P., ed., *Proceedings of the International Conference on Logic Programming (ICLP)*, volume 2401 of *LNCS*, 70–84.

Tuan, L.; Baral, C.; Zhang, X.; and Son, T. 2004. Regression With Respect to Sensing Actions and Partial States. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 556–561. AAAI Press.

Weld, D.; Anderson, C.; and Smith, D. 1998. Extending graphplan to handle uncertainty and sensing actions. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press.