# An Extension To Conformant Planning Using Logic Programming

**A. Ricardo Morales**
Texas Tech University
Computer Science Department
ricardo@cs.ttu.edu

**Phan Huy Tu** and **Tran Cao Son**
New Mexico State University
Department of Computer Science
{tphan,tson}@cs.nmsu.edu

## Abstract

In this paper we extend the logic programming based conformant planner described in [Son *et al.*, 2005a] to allow it to work on planning problems with more complex descriptions of the initial states. We also compare the extended planner with other concurrent conformant planners.

## 1 Introduction

Previous work [Gelfond and Morales, 2004; Son *et al.*, 2005b; 2005a; Tu *et al.*, 2006] have explored the usefulness of approximations of $\mathcal{AL}$ actions theories to conformant planning. Among them, [Son *et al.*, 2005a; Tu *et al.*, 2006] define different approximations in terms of logic programs and use these approximations to develop conformant planners in the logic programming paradigm. The choice of logic programming in the implementation of planners in general and conformant planners in particular offers several advantages. First, the correctness of the planner can be formally proved. Second, logic programming allows us to deal with domains with static causal laws (also called constraints). And finally, concurrent planning and domain control knowledge can be implemented in the planner without any difficulty.

The planners in [Son *et al.*, 2005a; 2005b; Tu *et al.*, 2006], however, suffer from two main drawbacks: (*i*) they work only on planning problems where the initial situation can be characterized by a single set of fluent literals; and (*ii*) they are in general incomplete, i.e., they might not be able to solve some problems which are solvable by other planners. Since these planners are based on an approximation theory, the latter problem is inevitable. It remained unclear, however, whether or not these planners can be extended to deal with other types of conformant planning problems, e.g. planning problems where the set of possible initial states is given by a disjunction. Furthermore, it is important to characterize conditions for which they are complete. An interesting work on this issue has recently been studied in [Son and Tu, 2006]. Nevertheless, the proposed completeness condition is restricted to action theories without static causal laws.

In this paper we address these limitations by (*a*) expanding the logic programming implementation of [Son *et al.*, 2005a] to build a new conformant planner, called CPASP$_m$, which can operate on planning problems with a more complex description of the initial state of the domain; and (*b*) extending the completeness result in [Son and Tu, 2006] to action theories with static causal laws.

This paper is organized as follows. In the next section, we review the basics of the action description language $\mathcal{AL}$ from [Baral and Gelfond, 2000] and the definition of an approximation from [Son *et al.*, 2005a]. In Section 3, we describe an extended logic programming based conformant planner called CPASP$_m$. In Section 4 we state the soundness of the implementation and provide a sufficient condition on action theories for which the planner is complete. Finally, in Section 5, we compare the performance of CPASP$_m$ with some other concurrent conformant planners.

## 2 Background

### 2.1 $\mathcal{AL}$ Action Theories and Transition Diagrams

Dynamic systems can be modeled by transition diagrams whose nodes correspond to possible states of the domain and whose arcs correspond to actions that take the domain from one state to another. We limit our attention to transition diagrams that can be defined by $\mathcal{AL}$ action theories.

The signature $\Sigma$ of an $\mathcal{AL}$ action theory consists of two disjoints sets of symbols: a set $\mathbf{F}$ of fluents and a set $\mathbf{A}$ of elementary actions. An *action* is a non-empty set of elementary actions. A *fluent literal* (or literal for short) is either a fluent $F$ or its negation $\neg F$. An $\mathcal{AL}$ action theory is a collection of statements of the forms:

$$e \textbf{ causes } l \textbf{ if } p \quad (1)$$

$$l \textbf{ if } p \quad (2)$$

$$\textbf{impossible } a \textbf{ if } p \quad (3)$$

where $e$ is an elementary action, $a$ is an action, and $p$ is a set of fluent literals from the signature $\Sigma$. Statement (1), called a *dynamic causal law*, says that, if $e$ is executed in a state satisfying $p$ then $l$ will hold in any resulting state. Statement (2), called a *static causal law*, says that any state satisfying $p$ must satisfy $l$. Statement (3) is an *impossibility condition*, which says that action $a$ cannot be performed in any state satisfying $p$. Given an action theory $\mathcal{D}$, a set of literals $s$ is *closed* under a static causal law (2) if $l \in s$ whenever $p \subseteq s$. For a set of literals $s$, by $Cn(s)$, we denote the smallest set of literals that includes $s$ and is closed under the set of static

causal laws of $\mathcal{D}$. A *state* $\sigma$ is a complete and consistent sets of literals closed under the set of static causal laws of $\mathcal{D}$. An action $b$ is said to be *prohibited* in state $\sigma$ if $\mathcal{D}$ contains an impossibility condition (3) s.t. $p \subseteq \sigma$ and $a \subseteq b$. For an action $a$ and a state $\sigma$, $E(a, \sigma)$ stands for the set of all literals $l$ s.t. $\mathcal{D}$ contains a law (1) with $p \subseteq \sigma$ and $e \in a$. Elements of $E(a, \sigma)$ are called *direct effects* of the execution of $a$ in $\sigma$. An action theory $\mathcal{D}$ describes a transition diagram $T(\mathcal{D})$. The nodes of $T(\mathcal{D})$ are states of $\mathcal{D}$ and arcs are labeled with actions. The transitions of $T(\mathcal{D})$ are defined as follows.

**Definition 1 ([McCain and Turner, 1997])** For an action $a$ and two states $\sigma$ and $\sigma'$, a transition $\langle \sigma, a, \sigma' \rangle \in T(\mathcal{D})$ iff $a$ is not prohibited in $\sigma$ and

$$\sigma' = Cn(E(a, \sigma) \cup (\sigma \cap \sigma')) \quad (4)$$

Given a state $\sigma$ and an action $a$, such $\sigma'$ is often referred to as a *possible successor state* of $\sigma$ as the result of the execution of $a$. When $\sigma$ and $a$ are clear from the context, we simply say that $\sigma'$ is a possible successor state.

An action theory $\mathcal{D}$ is *consistent* if for any state $\sigma$ and action $a$ which is not prohibited in $\sigma$ there is at least one possible successor state. In this paper, we consider consistent action theories only.

We now introduce some terminology used to describe properties of a transition diagram $T(\mathcal{D})$. By convention, we use letters $a$ and $\sigma$ (possibly indexed) to denote actions and states respectively. A *model* $M$ of a chain of events $\alpha = \langle a_0, \ldots, a_{n-1} \rangle$ is an alternate sequence of states and actions $\langle \sigma_0, a_0, \sigma_1, \ldots, a_{n-1}, \sigma_n \rangle$ s.t. $\langle \sigma_i, a_i, \sigma_{i+1} \rangle \in T(\mathcal{D})$ for $0 \leq i < n$. $\sigma_0$ (resp. $\sigma_n$) is referred to as the *initial state* (resp. *final state*) of $M$. We say that $M$ *entails* a set of literals $s$, written as $M \models s$, if $s \subseteq \sigma_n$. Furthermore, we sometimes write $\langle \sigma_0, \alpha, \sigma_n \rangle \in T(\mathcal{D})$ to denote that there is a model of $\alpha$ whose initial state and final state are $\sigma_0$ and $\sigma_n$, respectively.

An action $a$ is *executable* in a state $\sigma$ if there is a state $\sigma'$ s.t. $\langle \sigma, a, \sigma' \rangle \in T(\mathcal{D})$. A chain of events $\alpha = \langle a_0, \ldots, a_{n-1} \rangle$ is executable in $\sigma$ if either (i) $n = 0$ (i.e., $\alpha$ is an empty chain of events), or (ii) $a_0$ is executable in $\sigma$ and for every $\sigma'$ s.t. $\langle \sigma, a, \sigma' \rangle \in T(\mathcal{D})$, $\langle a_1, \ldots, a_{n-1} \rangle$ is executable in $\sigma'$.

## 2.2 Approximations of $T(\mathcal{D})$ and Conformant Planning Problem

In many of real world applications, an agent may not have complete knowledge of the current state of the domain. Under such circumstances it is not possible for the agent to directly reason about the effects of actions using the transition diagram of the domain. Our approach to reasoning with incomplete information is based on using approximations of action theories [Son and Baral, 2001; Son *et al.*, 2005b; 2005a; Tu *et al.*, 2006]. In this section, we review the definition of approximation from [Son *et al.*, 2005a] and discuss how an approximation can be used to find a solution of a conformant planning problem.

Let $\mathcal{D}$ be an action theory. A set $s$ of literals is a *partial state* if it is a subset of some state and it is closed under the set of static causal laws of $\mathcal{D}$. For a partial state $s$, a *completion* of $s$ is a state $\sigma$ s.t. $s \subseteq \sigma$ and by $comp(s)$ we denote the set of all completions of $s$. For a set of partial states $S$ by $comp(S)$ we denote the set $\bigcup_{s \in S} comp(s)$.

**Definition 2 (Approximation [Son *et al.*, 2005a])** $T'(\mathcal{D})$ is an *approximation* of $\mathcal{D}$ if:

1. Nodes of $T'(\mathcal{D})$ are partial states of $T(\mathcal{D})$.
2. If $\langle s, a, s' \rangle \in T'(\mathcal{D})$ then for every $\sigma \in comp(s)$,
   (a) $a$ is executable in $\sigma$, and
   (b) $s' \subseteq \sigma'$ for every $\sigma'$ s.t. $\langle \sigma, a, \sigma' \rangle \in T(\mathcal{D})$.

It is not difficult to see that the following observation holds.

**Observation 1 ([Son *et al.*, 2005a])** *Let $T'(\mathcal{D})$ be an approximation of $\mathcal{D}$. Then, for every chain of events $\alpha$, if $\langle s, \alpha, s' \rangle \in T'(\mathcal{D})$ then for every $\sigma \in comp(s)$, (a) $\alpha$ is executable in $\sigma$; and (b) $s' \subseteq \sigma'$ for every $\sigma'$ s.t. $\langle \sigma, \alpha, \sigma' \rangle \in T(\mathcal{D})$.*

This observation states that reasoning using approximations is sound wrt reasoning using $T(\mathcal{D})$.

One application of reasoning with partial states is conformant planning. In a conformant planning problem, the task of the agent is to find a chain of events that achieves some given goal provided it may not have complete knowledge of the initial state of the domain. More specifically,

**Definition 3** A *conformant planning problem* $\mathcal{P}$ is a tuple $\langle \mathcal{D}, \Gamma, G \rangle$ where $\Gamma$ is a set of possible initial states and $G$ is a set of literals.

**Definition 4** A chain of events $\alpha = \langle a_0, \ldots, a_{n-1} \rangle$ is a *solution* or a *conformant plan* of a conformant planning problem $\mathcal{P} = \langle \mathcal{D}, \Gamma, G \rangle$ if, for every $\sigma \in \Gamma$, $\alpha$ is executable in $\sigma$ and for every model $M$ of $\alpha$ with initial state $\sigma$, $M \models G$.

The following relationship between conformant plans and paths of an approximation of $\mathcal{D}$ follows from Observation 1.

**Observation 2** *Let $T'(\mathcal{D})$ be an approximation. If $\langle s, \alpha, s' \rangle \in T'(\mathcal{D})$ and $G \subseteq s'$ then $\alpha$ is a solution of the conformant planning problem $\langle \mathcal{D}, comp(s), G \rangle$.*

Observation 2 shows us a way to solve a conformant planning problem $\langle \mathcal{D}, \Gamma, G \rangle$: look for a path in an approximation that leads to a goal partial state (i.e. a partial state satisfying the goal). This approach is adopted in [Son *et al.*, 2005b; 2005a], where the set $\Gamma$ of possible initial states of a conformant planning problem is modeled as the completions of a single partial state $s$. It is shown in [Son *et al.*, 2005b; 2005a] that many conformant planning problems in the literature can be successfully solved this way. However, the following is an example of a conformant planning problem that cannot be solved by this method.

**Example 1 (Turkey and two guns)** There are two guns and a turkey. Initially only one of the guns is loaded and we do not know which one it is. The objective is to kill the turkey by shooting it. Let $g_1$ and $g_2$ be names for the guns and $X$ be a variable for the guns. This planning problem can be modeled as $\mathcal{P}_1 = \langle \mathcal{D}_1, \Gamma_1, G_1 \rangle$ where

$$\mathcal{D}_1 = \left\{ \begin{array}{l} shoot(X) \textbf{ causes } dead \textbf{ if } loaded(X) \\ shoot(X) \textbf{ causes } \neg loaded(X) \end{array} \right\}$$

$$\Gamma_1 = \left\{ \begin{array}{l} \{\neg dead, loaded(g_1), \neg loaded(g_2)\}, \\ \{\neg dead, \neg loaded(g_1), loaded(g_2)\} \end{array} \right\}$$

$$G_1 = \{dead\}$$

It is easy to see that by Definition 4 $\alpha_1 = \langle\{shoot(g_1)\}, \{shoot(g_2)\}\rangle$ is a solution of $\mathcal{P}_1$.

Following the approach in [Son *et al.*, 2005b], however, the conformant planning is modeled as $\mathcal{P}_1' = \langle\mathcal{D}_1, \Gamma_1', \{dead\}\rangle$ where $\Gamma_1' = comp(\{\neg dead\})$. Clearly $\mathcal{P}_1'$ does not have a conformant plan because the state $\sigma = \{\neg dead, \neg loaded(g_1), \neg loaded(g_2)\}$ belongs to $comp(\{\neg dead\})$ and there exists no path in the transition diagram $T(\mathcal{D})$ that leads to a goal state from $\sigma$. □

In order to address this issue, we have extended the definition of a conformant plan from [Son *et al.*, 2005a], and, with a more careful examination of approximations, conclude the following observation, which can be viewed as a generalization of Observation 2.

**Observation 3** *Let $\mathcal{P} = \langle\mathcal{D}, \Gamma, G\rangle$ be a planning problem, $T'(\mathcal{D})$ be an approximation, and $\alpha$ be a chain of events. Let $S$ be a set of partial states s.t. $comp(S) = \Gamma$. If for every $s \in S$, there exists a partial state $s'$ s.t. $\langle s, \alpha, s'\rangle \in T'(\mathcal{D})$ and $G \subseteq s'$ then $\alpha$ is a solution of $\mathcal{P}$.*

This observation extends the applicability of approximations to conformant planning problems with a more complex description of the initial state of the domain. In particular, instead of viewing the set of possible initial states $\Gamma$ as (the completions of) a single initial partial state $s$ as in [Son *et al.*, 2005a], we view it as a set of initial partial states $S$. Then, to find a conformant plan within an approximation, we look for a chain of events $\alpha$ s.t. every possible path of $\alpha$ starting from a partial state in $S$ always leads to a goal partial state. As an example, the set of possible initial states of the domain in Example 1 can be viewed as the set $S_1$ of two partial states[1] $\{\neg dead, loaded(g_1), \neg loaded(g_2)\}$ and $\{\neg dead, \neg loaded(g_1), loaded(g_2)\}$, i.e., $S_1 = \Gamma_1$. It is easy to verify that, within the approximation in [Son *et al.*, 2005a], the solution $\alpha_1 = \langle shoot(g_1), shoot(g_2)\rangle$ can be found by this method because every possible path of $\alpha_1$ in the approximation that starts from a partial state in $S_1$ always leads to a goal partial state.

## 3 A Logic Programming Implementation

We now describe an answer set planning [Subrahmanian and Zaniolo, 1995] based method for constructing an approximation to action theories of $\mathcal{AL}$. In our method, we define a transition diagram $\tilde{T}(\mathcal{D})$ in terms of a logic program $\tilde{\pi}(\mathcal{D})$ called the *cautious encoding* of $\mathcal{D}$. The work in [Son *et al.*, 2005a] uses a similar construction but the resulting logic program is only able to deal with a single initial partial state. The work in [Tu *et al.*, 2006] does provide a method for constructing conditional plans with several branches but operates from a single initial partial state as well. Our work in this paper can be viewed as a combination of the ideas presented in these works in order to develop a conformant planner which can deal with a more complex description of the initial state; in particular, the planner can work on domains with disjunctive information in the initial state. In order to achieve this, a distinct label is attached to every partial state considered. Atoms of $\tilde{\pi}(\mathcal{D})$ are formed by the following (sorted) predicate symbols:

- $h(l, t, k)$ is true if literal $l$ holds at time step $t$ in a path starting from the initial partial state labeled with $k$;
- $o(e, t)$ is true if $e$ occurs at time step $t$;
- $dc(l, t, k)$ is true if literal $l$ is a direct effect of an action that occurs at $(t-1, k)$; and
- $ph(l, t, k)$ is true if literal $l$ possibly holds at time $t$ in a path with the initial partial state labeled with $k$.

The program also contains auxiliary predicates: $time$, $fluent$, $action$, and $label$ that are used for enumerating constants of sorts time step, fluent, elementary action and label respectively. In our presentation, we also use some shorthands: if $a$ is an action then $o(a, t) = \{o(e, t) : e \in a\}$. If $p$ is a set of literals, and $\mathcal{F}$ is either $h$, $dc$, or $ph$, then $\mathcal{F}(p, t, k) = \{\mathcal{F}(l, t, k) : l \in p\}$ and $not\ \mathcal{F}(p, t, k) = \{not\ \mathcal{F}(l, t, k) : l \in p\}$. For a literal $l$, by $\overline{l}$ we denote its complement, i.e. $\bar{F} = \neg F$ and $\overline{\neg F} = F$. For a set of literal $p$, $\overline{p} = \{\overline{l} : l \in p\}$. Finally, $L$, $T$ and $K$ (possibly with indices) are variables for literals, time steps, and labels respectively. For convenience, we often write $\tilde{\pi}(\mathcal{D}, n)$ to denote the restriction of $\tilde{\pi}(\mathcal{D})$ to time steps between 0 and $n$. The set of rules[2] of $\tilde{\pi}(\mathcal{D})$ is listed below.

1. For each dynamic causal law (1) in $\mathcal{D}$, the rules

$$h(l, T+1, K) \leftarrow o(e, T), h(p, T, K) \quad (5)$$
$$dc(l, T+1, K) \leftarrow o(e, T), h(p, T, K) \quad (6)$$

belong to $\tilde{\pi}(\mathcal{D})$. The first rule states that $l$ holds at $T+1$ on path $K$ if $e$ occurs at $T$ and precondition $p$ holds at $T$ on path $K$. The second rule states that $l$ is a direct effect of $e$ at $(T+1, K)$ if $e$ occurs at $T$ and precondition $p$ holds at $(T, K)$.

Since the state of the domain at time step $T$ might be incomplete, we add to $\tilde{\pi}(\mathcal{D})$ the rule

$$ph(l, T+1, K) \leftarrow o(e, T),\ not\ h(\overline{p}, T, K) \quad (7)$$

which says that $l$ possibly holds at $(T+1, K)$ if $e$ occurs at $T$ and the precondition $p$ possibly holds at $(T, K)$.

Furthermore, we add to $\tilde{\pi}(\mathcal{D})$ the following rule

$$ph(L, T+1, K) \leftarrow not\ h(\overline{L}, T, K),\ not\ dc(\overline{L}, T+1, K). \quad (8)$$

which says that a literal $L$ possibly holds at $T+1$ if it possibly holds at $T$ and its negation is not a direct effect of the action that occurs at $T$.

2. For each static causal law (2) in $\mathcal{D}$, $\tilde{\pi}(\mathcal{D})$ contains:

$$h(l, T, K) \leftarrow h(p, T, K) \quad (9)$$
$$ph(l, T, K) \leftarrow ph(p, T, K) \quad (10)$$

These rules state that if $p$ holds (or possibly holds) at $(T, K)$ then so does $l$.

[1] Note that by definition a state is also a partial state.

[2] In each rule, predicates defining sorts of variables are omitted for brevity.

3. For each impossibility condition (3) in $\mathcal{D}$, we add to $\tilde{\pi}(\mathcal{D})$ the following rule:

$$\leftarrow \quad o(a,T),\ not\ h(\overline{p},T,K) \qquad (11)$$

This rule states that $a$ cannot occur at time step $T$ if precondition $p$ possibly holds at $(T,K)$ for some path $K$.

4. The inertial law is encoded as follows:

$$h(L,T,K) \quad \leftarrow \quad not\ ph(\overline{L},T,K),\ T \neq 0. \quad (12)$$

which says that $L$ holds at time step $T > 0$ if its negation cannot possibly hold at $T$.

5. *Auxiliary rule:* $\tilde{\pi}(\mathcal{D})$ also contains the following rule:

$$\leftarrow \quad h(F,T,K), h(\neg F,T,K) \quad (13)$$

$$literal(F) \quad \leftarrow \quad fluent(F) \qquad (14)$$

$$literal(\neg F) \quad \leftarrow \quad fluent(F) \qquad (15)$$

The first rule, as a constraint, states that $h(f,t,k)$ and $h(\neg f,t,k)$ cannot hold at the same time [3] . The last two rules are used to define literals.

For an action $a$ and a partial state $s$, let $\Pi(\mathcal{D},a,s) = \tilde{\pi}(\mathcal{D},1) \cup h(s,0,0) \cup o(a,0)$. Note that $\Pi(a,s)$ is a stratified program, so if it is consistent, then it has a unique answer set [Gelfond and Lifschitz, 1988]. We define an approximation $\tilde{T}(\mathcal{D})$ based on $\Pi(\mathcal{D},a,s)$ as follows.

**Definition 5** Let $s$ be a partial state and $a$ be an action. Let $\tilde{T}(\mathcal{D})$ be a transition diagram s.t. $\langle s,a,s'\rangle \in \tilde{T}(\mathcal{D})$ iff $\Pi(\mathcal{D},a,s)$ is consistent and $s' = \{l : h(l,1,0) \in A\}$ where $A$ is the answer set of $\Pi(\mathcal{D},a,s)$.

**Theorem 1** If $\mathcal{D}$ is consistent then $\tilde{T}(\mathcal{D})$ is an approximation of $\mathcal{D}$. □

The proof of this theorem is similar to the proofs of the soundness of the logic programming encoding of $\mathcal{AL}$ action theories in [Son *et al.*, 2006; Turner, 1997].

Let $\mathcal{P} = \langle \mathcal{D},\Gamma,G\rangle$ be a conformant planning problem. Since $\tilde{T}(\mathcal{D})$ is an approximation of $\mathcal{D}$, we can use the program $\tilde{\pi}(\mathcal{D})$ to find solutions of $\mathcal{P}$. In the remaining part of this section, we describe a logic program $\tilde{\pi}(\mathcal{P},n)$, where $\mathcal{P}$ is the input planning problem and $n$ is the length of a solution we wish to find.

Let $S = \{s_0,s_1,..,s_m\}$ be a set of partial states s.t. $\Gamma = comp(S)$ (hereafter we refer to such $S$ as a set of initial partial states of $\mathcal{P}$). Let the subindex of each element of $S$ act as its label. We construct the program $\tilde{\pi}(\mathcal{P},n)$ based on $S$ as follows. $\tilde{\pi}(\mathcal{P},n)$ contains the rules of $\tilde{\pi}(\mathcal{D},n)$ and the following rules.

1. *Rules encoding the initial state:* For every $s_i \in S$, $\tilde{\pi}(\mathcal{P},n)$ contains the rule

$$h(s_i,0,i). \qquad (16)$$

2. *Rules encoding the goal:* To guarantee that the goal is satisfied in the final state of every path, we add to $\tilde{\pi}(\mathcal{P},n)$ the following rules:

$$\{\leftarrow not\ h(l,n,K) : l \in G\}.$$

3. *Rules to generate actions:* To generate action occurrences, we add to $\tilde{\pi}(\mathcal{P},n)$ the following choice rule [Simons *et al.*, 2002] [4]:

$$1\{o(E,T) : action(E)\} \quad \leftarrow \quad T < n \qquad (17)$$

## 4 Properties of $\tilde{\pi}(\mathcal{P},n)$

In this section, we investigate the soundness and completeness of $\tilde{\pi}(\mathcal{P},n)$.

### 4.1 Soundness

Using Theorem 1 and Observation 3, we can prove the correctness of the planner $\tilde{\pi}(\mathcal{P},n)$.

**Theorem 2** Let $A$ be an answer set of $\tilde{\pi}(\mathcal{P},n)$. For $0 \leq i < n$ let $a_i = \{e : o(e,i) \in A\}$. Then, $\alpha = \langle a_0,..,a_{n-1}\rangle$ is a solution of $\mathcal{P}$. □

### 4.2 A Sufficient Condition for the Completeness

Although Theorem 2 tells us that $\tilde{\pi}(\mathcal{P},n)$ is sound, the program is not complete in the sense that there are planning problems that have solutions for which $\tilde{\pi}(\mathcal{P})$ returns no answer set. This is illustrated in the following example.

**Example 2** Consider the action description

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

and the planning problem $\mathcal{P}_2 = \langle \mathcal{D}_2,\Gamma_2,\{g\}\rangle$ where $\Gamma_2$ is the set of all possible states. Obviously $\langle a\rangle$ is a solution of this problem because either $f$ or $\neg f$ is true in any state belonging to $\Gamma_2$ and thus either one of the above dynamic causal laws would take effect when $a$ is executed.

Let $S_1 = \{\emptyset\}$. Then, it is easy to see that $comp(S_1) = \Gamma_2$. If we construct $\tilde{\pi}(\mathcal{P}_2,1)$ based on $S_1$ then $\tilde{\pi}(\mathcal{P}_2,1)$ is inconsistent, which means that no conformant plan of length one can be found by $\tilde{\pi}(\mathcal{P}_2,1)$. In fact, it can be shown that for every $n$, $\tilde{\pi}(\mathcal{P}_2,n)$ is inconsistent. Thus, $\tilde{\pi}(\mathcal{P}_2,n)$ cannot solve the planning problem $\mathcal{P}_2$. However, if $\tilde{\pi}(\mathcal{P}_2,1)$ is constructed based on the set of initial partial states $S_2 = \{\{f\},\{\neg f\}\}$ then $\tilde{\pi}(\mathcal{P}_2,1)$ would return an answer set corresponding to the solution $\langle a\rangle$. □

The above example shows that choosing a set of initial partial states $S$ plays an important role in the completeness of $\tilde{\pi}(\mathcal{P},n)$. To characterize the situation in Example 2, we define a notion of dependencies among literals and between literals and actions (adapted from [Son and Tu, 2006]).

**Definition 6** A literal $l$ *depends on* a literal $g$, written as $l \triangleleft g$, whenever one of the following conditions holds.

1. $l = g$.
2. $\mathcal{D}$ contains a law $[a$ **causes** $l$ **if** $p]$ s.t. $g \in p$.
3. $\mathcal{D}$ contains a law $[l$ **if** $p]$ s.t. $g \in p$.
4. There exists a literal $h$ s.t. $l \triangleleft h$ and $h \triangleleft g$.
5. $\overline{l} \triangleleft \overline{g}$.

[3] Some adjustment to this syntax would be needed if one wants to use an existing answer set solvers. For instance, we may need to replace $\neg f$ with, say, $neg(f)$.

[4] If we wish to find a sequential plan, the only thing needed to do is to change the left side of the rule to $1\{o(E,T) : action(E)\}1$.

**Definition 7** An action $b$ *depends on* a literal $l$, written as $b \triangleleft l$, if either

1. $\mathcal{D}$ contains an impossibility condition (3) s.t. $a \subseteq b$ and $\bar{l} \in \psi$, or
2. there exists a literal $g$ s.t. $b \triangleleft g$ and $g \triangleleft l$.

For a set of literals $w$, we write $l \triangleleft w$ to denote that $l \triangleleft g$ for some $g \in w$, and $l \not\triangleleft w$ to denote that there exists no $g \in w$ s.t. $l \triangleleft g$.

Observe that according to Definition 6, for the domain $\mathcal{D}_2$ in Example 2, we have that $g \triangleleft f$ and $g \triangleleft \neg f$. This implies that for each state $\sigma$ in $\Gamma_2$, we can find some literal $l$ such that $g \triangleleft l$. In this case, we say that $\Gamma_2$ is not reducible to $\emptyset$ wrt $\{g\}$. The precise definition of reducibility is given below.

**Definition 8** *Let $\Delta$ be a set of states, $s$ be a partial state, and $w$ be a set of literals. We say that $\Delta$ is* reducible *to $s$ wrt $w$, denoted by $\Delta \gg_w s$ if*

1. *$s$ is a subset of every state $\sigma$ in $\Delta$,*
2. *for each $l \in w$, $\Delta$ contains a state $\sigma$ s.t. $l \not\triangleleft (\sigma \setminus s)$, and*
3. *for each action $a$, $\Delta$ contains a state $s$ s.t. $a \not\triangleleft (\sigma \setminus s)$.*

We say that a set of states $\Delta$ is reducible to a set of partial states $S$ wrt a set of literals $w$, written as $\Delta \gg_w S$ iff $\Delta = comp(S)$ and $comp(s) \gg_w s$ for every $s \in S$. Back to Example 2, according to this definition, we do not have $\Gamma_2 \gg_{\{g\}} S_1 = \{\emptyset\}$ but we do have $\Gamma_2 \gg_{\{g\}} S_2 = \{\{f\}, \{\neg f\}\}$.

We define a class of action theories as follows.

**Definition 9** A static causal law is *simple* if its precondition contains at most one literal. An action theory $\mathcal{D}$ is *simple* if each of its static causal laws is simple.

We notice that if $\mathcal{D}$ is simple then $\Delta \gg_w S$ implies that reasoning with $\tilde{T}(\mathcal{D})$ from the set of partial states $S$ would agree with reasoning about $w$ with $T(\mathcal{D})$ from the set of possible initial states $\Delta$ on the set of literals $w$. Base on this observation, we have the following result about the completeness of $\tilde{\pi}(\mathcal{P}, n)$.

**Theorem 3** Let $\alpha = \langle a_1, \ldots, a_{n-1} \rangle$ be a solution of $\mathcal{P} = \langle \mathcal{D}, \Gamma, G \rangle$. If $\mathcal{D}$ is simple and $\tilde{\pi}(\mathcal{P}, n)$ is constructed based on a set of initial partial states $S$ s.t. $\Gamma \gg_G S$ then $\tilde{\pi}(\mathcal{P}, n)$ is consistent and has an answer set $A$ s.t. $a_i = \{e : o(e, i) \in A\}$. □

We can verify that for the planning problem $\mathcal{P}_2$ in Example 2, if we construct $\tilde{\pi}(\mathcal{P}, n)$ based on $S_1 = \{\emptyset\}$ then we would miss the solution $\langle a \rangle$. However, if we construct it based on the set of partial states $S_2 = \{\{f\}, \{\neg f\}\}$ then $\tilde{\pi}(\mathcal{P}, 1)$ would return an answer set corresponding to the solution $\langle a \rangle$.

## 5 Experiments

Theorem 2 suggests a simple algorithm for finding minimal plans (wrt the number of steps) of a planning problem $\mathcal{P}$: sequentially run program $\tilde{\pi}(\mathcal{P}, n)$ with $n = 1, 2, \ldots$, until it returns an answer set (we assume that $\mathcal{P}$ does not have a trivial solution $\langle\rangle$). We have implemented this algorithm in a planning system called CPASP$_m$ and use CMODELS [Lierler and Maratea, 2004] as the underlying answer set solver.

| No. Guns | PL | DLV$^{\text{K}}$ | $\mathcal{C}$-PLAN | CPASP$_m$ |
|---|---|---|---|---|
| 2 | 2 | 0.03 | 0.036 | 0.129 |
| 4 | 3 | 0.11 | 0.044 | 0.233 |
| 6 | 3 | 0.72 | 0.186 | 0.296 |
| 8 | 3 | 11.58 | 2.172 | 0.388 |
| 10 | 3 | 1039.96 | 50.201 | 0.503 |
| 20 | 3 | - | - | 1.487 |
| 50 | 3 | - | - | 8.208 |
| 100 | 3 | - | - | 32.623 |

Table 1: Conformant Turkey domain

CPASP$_m$ takes as input a logic program representation of $\mathcal{P}$ and produces concurrent conformant plans. We have compared CPASP$_m$ with other conformant planning systems. Due to space limit we only consider planners capable of generating concurrent plans. As aforementioned, only a small modification to the logic program can make the planner act as a sequential planner.

We compare CPASP$_m$ with the concurrent conformant planners DLV$^{\text{K}}$ [Eiter *et al.*, 2003] and $\mathcal{C}$-PLAN [Castellini *et al.*, 2003]. $\mathcal{C}$-PLAN is a SAT-based conformant planner where a planning problem is translated into a satisfiability problem and satisfying assignments of variables correspond to solutions of the original planning problem. DLV$^{\text{K}}$ is somewhat similar to CPASP$_m$ as both run on top of an answer set solver. However, unlike CPASP$_m$ DLV$^{\text{K}}$ does not use approximations. Its algorithm has two phases: looking for candidate plans first and then checking if they are conformant.

We prepared a test suite of problems which involves both concurrency and complex initial situations. Some of this problems are simple modifications to conformant benchmarks in the literature. The test suite consists of the problems *Conformant Turkey* and *Lost Cleaner*. *Conformant Turkey* is a modification of the classic Yale shooting problem. In our problem, there are $n$ guns, but only one of which is loaded. Guns can be shot concurrently, with the exception of gun #1 and #2. The objective is to kill the turkey. *Lost Cleaner* is a modification of the *ring* [Cimatti *et al.*, 2004]. The agent is in one of $n$ rooms arranged in a ring fashion. Each room has $m$ objects that must be cleaned by the agent. The agent can move to the next room in a clockwise or counter-clockwise fashion and at a time it can clean objects in a room concurrently. Initially, the agent does not know its initial location nor the current status (cleaned or dirty) of the objects.

Experiments were conducted in a 1.4GHz Pentium 4 machine running Linux, and each planner was given 30 minutes to complete each instance of each problem. The results are shown in Tables 1 and 2. In each table, times are in seconds, and $PL$ is the number of steps of a shortest solution of the corresponding problem.

As we can see from Table 1, CPASP$_m$ outperforms both DLV$^{\text{K}}$ and $\mathcal{C}$-PLAN in the *Conformant Turkey* domain. Solutions for these domain all involve 3 steps, with one step consisting of firing multiple guns concurrently. In this domain, exploiting concurrency is key to obtaining a short plan.

In the *Lost Cleaner* domain, shown in Table 2, only CPASP$_m$ is able to solve all problem instances within the time limit. In this domain, the length of a solution ranges from 3 to 19 steps and it is determined by the number of rooms in

| No. Rooms | No. Obj. | PL | DLV$^{\mathcal{K}}$ | $\mathcal{C}$-PLAN | CPASP$_m$ |
|---|---|---|---|---|---|
| 2 | 2 | 3 | 0.068 | 0.063 | 0.201 |
| 2 | 5 | 3 | 0.092 | - | 0.233 |
| 2 | 10 | 3 | 0.143 | - | 0.288 |
| 4 | 2 | 7 | 0.483 | 425.048 | 1.219 |
| 4 | 5 | 7 | 2.648 | - | 1.897 |
| 4 | 10 | 7 | 441.485 | - | 3.015 |
| 6 | 2 | 11 | 35.901 | - | 7.190 |
| 6 | 5 | 11 | - | - | 12.414 |
| 6 | 10 | 11 | - | - | 22.503 |
| 8 | 2 | 15 | - | - | 51.147 |
| 8 | 5 | 15 | - | - | 76.707 |
| 8 | 10 | 15 | - | - | 142.563 |
| 10 | 2 | 19 | - | - | 225.405 |
| 10 | 5 | 19 | - | - | 403.443 |
| 10 | 10 | 19 | - | - | 642.319 |

Table 2: Lost Cleaner Domain

the ring. As the number of rooms in the problem increases, the number of possible initial situations increases. One of the advantages of using approximations is that it is possible to reason about the domain without having to consider all possible worlds. As the complexity of the problem increases, only CPASP$_m$ is able to scale. In this problem $\mathcal{C}$-PLAN behaved erratically, failing some simple problem instances and then solving more complicated ones. In general, the use of approximations allows CPASP$_m$ to exploit concurrent actions when searching for a solution, and to scale better than the other concurrent planners as the number of possible initial states and the length of the solution increases.

## 6 Conclusion and Future Work

In this paper, we extend the logic programming based approach to conformant planning in [Son *et al.*, 2005a; Tu *et al.*, 2006] to develop a conformant planner, called CPASP$_m$, which can solve planning problems with a more complex description of the initial states. The planner inherits several key properties of the planner CPASP in [Son *et al.*, 2005a]. In particular, it can deal with static causal laws and is capable of finding minimal and concurrent plans. It is sound and sometimes complete wrt the full semantics of $\mathcal{AL}$ action theories. Our experimental results show that the planner is comparable with some other concurrent conformant planners.

As future work, one of our goals is to strengthen the condition for the completeness of the planner. Also, we would like to develop a method for including domain knowledge so that we can obtain better performance from the planner.

**Acknowledgment:** We will like to thank Dr. Michael Gelfond for his collaboration, help, and useful discussions on the topic.

## References

[Baral and Gelfond, 2000] C. Baral and M. Gelfond. Reasoning agents in dynamic domains. *Logic-Based Artificial Intelligence*, pages 257–279, 2000.

[Castellini *et al.*, 2003] C. Castellini, E. Giunchiglia, and A. Tacchella. SAT-based Planning in Complex Domains: Concurrency, Constraints and Nondeterminism. *AI*, 147:85–117, July 2003.

[Cimatti *et al.*, 2004] A. Cimatti, M. Roveri, and P. Bertoli. Conformant Planning via Symbolic Model Checking and Heuristic Search. *AI*, 159:127–206, 2004.

[Eiter *et al.*, 2003] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A Logic Programming Approach to Knowledge State Planning, II: The DLV$^{\mathcal{K}}$ System. *AI*, 144(1-2):157–211, 2003.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *ICLP*, pages 1070–1070, MIT Press, 1988.

[Gelfond and Morales, 2004] M. Gelfond and A.R. Morales. Encoding conformant planning in a-prolog. In *DRT*, LNCS, Springer, 2004.

[Lierler and Maratea, 2004] Y. Lierler and M. Maratea 2004. Cmodels-2: SAT-based Answer Set Solver Enhanced to Non-tight Programs. In *LPNMR*, pages 346–350, 2004.

[Lifschitz, 2002] V. Lifschitz. Answer set programming and plan generation. *AI*, 138(1–2):39–54, 2002.

[McCain and Turner, 1997] N. McCain and H. Turner. Causal theories of action and change. In *AAAI*, pages 460–467. AAAI Press, 1997.

[Simons *et al.*, 2002] P. Simons, N. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *AI*, 138(1–2):181–234, 2002.

[Subrahmanian and Zaniolo, 1995] V.S. Subrahmanian, C. Zaniolo. Relating Stable Models and AI Planning Domains. In *ICLP*, pages 233–247, MIT Press, 1995.

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

[Son and Tu, 2006] T. C. Son and P. H. Tu. On the Completeness of Approximation Based Reasoning and Planning in Action Theories with Incomplete Information. In *KR*, pages 481–491, 2006.

[Son *et al.*, 2005a] T. C. Son, P. H. Tu, M. Gelfond, and A.R. Morales. An Approximation of Action Theories of $\mathcal{AL}$ and its Application to Conformant Planning. In *LPNMR*, pages 172–184, 2005.

[Son *et al.*, 2005b] T. C. Son, P. H. Tu, M. Gelfond, and A.R. Morales. Conformant Planning for Domains with Constraints — A New Approach. In *AAAI*, pages 1211–1216, 2005.

[Son *et al.*, 2006] T. C. Son, C. Baral, N. Tran, and S. McIlraith. Domain-Dependent Knowledge in Answer Set Planning. *ACM TOCL*, 7(4), 2006.

[Tu *et al.*, 2006] P. H. Tu, T. C. Son, and C. Baral. Reasoning and Planning with Sensing Actions, Incomplete Information, and Static Causal Laws using Logic Programming. *TPLP*, 7:1–74, 2006.

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