A Correctness Result for Reasoning about One-Dimensional Planning Problems

Yuxiao (Toby) Hu   Hector J. Levesque

Department of Computer Science  
University of Toronto

{yuxiao,hector}@cs.toronto.edu

May 12, 2010
Motivation

- Classical planning produces action sequence in complete world.
  - e.g.: given, $obj1$ at home, $obj2$ in office and a $truck$, make $obj1$ in office and $obj2$ at home.
  - Resulting sequential plan only works for this particular setting.

Conditional planning allows incomplete knowledge by allowing branching on run-time world state.

- e.g.: given a truck, $obj1$ and $obj2$, location and destination unknown, make both objects at their destination.
- Resulting tree-like plan can handle four different cases.
- An even more general form of planning?
  - Given a truck and an unknown number of objects, make them all at their desired destination!
  - Incomplete knowledge about number results in infinitely many cases.
Motivation

- Classical planning produces action sequence in complete world.
  - *e.g.*: given, \textit{obj1} at home, \textit{obj2} in office and a \textit{truck}, make \textit{obj1} in office and \textit{obj2} at home.
  - Resulting sequential plan only works for this particular setting.

- Conditional planning allow incomplete knowledge by allowing branching on run-time world state.
  - *e.g.*: given a truck, \textit{obj1} and \textit{obj2}, location and destination unknown, make both objects at their destination.
  - Resulting tree-like plan can handle four different cases.
Motivation

- Classical planning produces action sequence in complete world.
  - e.g.: given, obj1 at home, obj2 in office and a truck, make obj1 in office and obj2 at home.
  - Resulting sequential plan only works for this particular setting.

- Conditional planning allow incomplete knowledge by allowing branching on run-time world state.
  - e.g.: given a truck, obj1 and obj2, location and destination unknown, make both objects at their destination.
  - Resulting tree-like plan can handle four different cases.

- An even more general form of planning?
  - Given a truck and an unknown number of objects, make them all at their desired destination!
  - Incomplete knowledge about number results in infinitely many cases.
Motivation

An intuitive plan:
Motivation

An intuitive plan:

Moral:
- With this generality, **plans with loops** are needed!
Outline of the Talk

1. Planning with Loops
2. A Formal Notion of Correctness
3. Practical Verification for 1-D Problems
4. Conclusion
FSAPLANNER (Hu & Levesque 09) generates plans with loops by

1. generating a plan with loops that works for small instances;
2. testing if the plan also works for other instances.
   (If not, return to Step 1.)
Planning with Loops

FSAPLANNER (Hu & Levesque 09) generates plans with loops by

1. generating a plan with loops that works for small instances;
2. testing if the plan also works for all (?) other instances. (If not, return to Step 1.)

Plan Verification...

- Ideally, a candidate plan may pass the testing phase, only if it works for all instances of the planning problem.
Planning with Loops

FSAPLANNER (Hu & Levesque 09) generates plans with loops by

1. generating a plan with loops that works for small instances;
2. testing if the plan also works for all (?) other instances. (If not, return to Step 1.)

Plan Verification...

- Ideally, a candidate plan may pass the testing phase, only if it works for all instances of the planning problem.
- However, this seems impossible with infinitely many cases.
FSAPLANNER (Hu & Levesque 09) generates plans with loops by
1. generating a plan with loops that works for small instances;
2. testing if the plan also works for some other instances.
   (If not, return to Step 1.)

Plan Verification...
- Ideally, a candidate plan may pass the testing phase, only if it works for all instances of the planning problem.
- However, this seems impossible with infinitely many cases.
- In practice, we only test against finitely many larger instances.
Planning with Loops

FSAPLANNER (Hu & Levesque 09) generates plans with loops by
1. generating a plan with loops that works for small instances;
2. testing if the plan also works for some other instances.
   (If not, return to Step 1.)

Plan Verification...

- Ideally, a candidate plan may pass the testing phase, only if it works for all instances of the planning problem.
- However, this seems impossible with infinitely many cases.
- In practice, we only test against finitely many larger instances.

Needed: finite verification with general correctness guarantee!
Contributions

In this paper, we

1. formally define a representation (FSA plan) for plans with loops;
2. identify a class of (one-dimensional) planning problems whose plan correctness can be finitely verified;
3. show that this verification algorithm enables FSAPLANNER to efficiently generate provably correct plans for this problem class.
The situation calculus is a multi-sorted logic for modeling dynamic environments, with sorts *situation*, *action* and *object*.

- $S_0$ is the unique initial situation, and $do(a, s)$ is the situation obtained by performing action $a$ in situation $s$.
- Changing properties modeled by fluents, *i.e.*, functions and predicates whose last argument is a situation term, *e.g.*, 

$$\text{loc}(S_0) = \text{home} \land \text{Loaded}(do(\text{load}, S_0)).$$

- $\text{Poss}(a, s)$ is a special relation that holds iff action $a$ is executable in situation $s$.
- $\text{SR}(a, s)$ denotes the sensing result of action $a$ when performed in situation $s$. 
Problem Representation

The dynamics of a planning problem is axiomatized by a Basic Action Theory (Reiter 01)

\[ \Sigma = \mathcal{FA} \cup \Sigma_{una} \cup \Sigma_{pre} \cup \Sigma_{ssa} \cup \Sigma_{sr} \cup \Sigma_0, \]
Problem Representation

The dynamics of a planning problem is axiomatized by a Basic Action Theory (Reiter 01)

\[ \sum = \mathcal{FA} \cup \sum_{una} \cup \sum_{pre} \cup \sum_{ssa} \cup \sum_{sr} \cup \sum_0, \]

where \( \sum_{sr} \) is a set of sensing result axioms (Scherl & Levesque 03):

\[ \text{SR}(\text{get}_\text{done}, s) = r \equiv r = \text{yes} \land \text{parcels}_\text{left}(s) = 0 \lor \]
\[
  r = \text{no} \land \text{parcels}_\text{left}(s) \neq 0.\]
The dynamics of a planning problem is axiomatized by a Basic Action Theory (Reiter 01)

\[ \Sigma = \mathcal{F}A \cup \Sigma_{una} \cup \Sigma_{pre} \cup \Sigma_{ssa} \cup \Sigma_{sr} \cup \Sigma_0, \]

where \( \Sigma_{sr} \) is a set of sensing result axioms (Scherl & Levesque 03):

\[ \text{SR}(\text{get done}, s) = r \equiv r = \text{yes} \land \text{parcels left}(s) = 0 \lor \]

\[ r = \text{no} \land \text{parcels left}(s) \neq 0. \]

**Definition**

A planning problem is a pair \( \langle \Sigma, G \rangle \), where \( \Sigma \) is a basic action theory and \( G \) is a situation-suppressed goal formula.
Problem Representation

The dynamics of a planning problem is axiomatized by a Basic Action Theory (Reiter 01)

\[ \Sigma = \mathcal{FA} \cup \Sigma_{una} \cup \Sigma_{pre} \cup \Sigma_{ssa} \cup \Sigma_{sr} \cup \Sigma_0, \]

where \( \Sigma_{sr} \) is a set of sensing result axioms (Scherl & Levesque 03):

\[ \text{SR}(\text{get\_done}, s) = r \equiv r = \text{yes} \land \text{parcels\_left}(s) = 0 \lor \]
\[ r = \text{no} \land \text{parcels\_left}(s) \neq 0. \]

**Definition**

A planning problem is a pair \( \langle \Sigma, G \rangle \), where \( \Sigma \) is a basic action theory and \( G \) is a situation-suppressed goal formula.

Both infinite domain and incomplete initial state allowed.
We use a finite-state automaton-like plan representation (called FSA plan), which can be viewed as a directed graph, where

- Each node represents a program state
  - One unique “start state”
  - One unique “final state”
  - Non-final states associated with action
- Each edge labeled with a sensing result (omitted for non-sensing).
Plan Representation

To formalize FSA plans, we introduce a new sort “program states” with $Q_0$ and $Q_F$ being two constants, and a set of axioms $FSA$, consisting of

1. domain closure axioms for program states

$$(\forall q).q = Q_0 \lor q = Q_1 \lor \cdots \lor q = Q_n \lor q = Q_F;$$

2. unique names axioms for program states

$$Q_i \neq Q_j \text{ for } i \neq j;$$

3. action association axioms

$$\gamma(Q) = A;$$

4. transition axioms

$$\delta(Q, R) = Q'.$$
Plan Correctness

We use $T(q, s, q', s')$ to denote legal one-step transitions, i.e.,

$$T(q, s, q', s') \overset{\text{def}}{=} \exists a, r. \gamma(q) = a \land \text{Poss}(a, s) \land \text{SR}(a, s) = r \land \delta(q, r) = q' \land s' = \text{do}(a, s)$$
Plan Correctness

We use $T(q, s, q', s')$ to denote legal one-step transitions, i.e.,

$$T(q, s, q', s') \overset{\text{def}}{=} \exists a, r. \gamma(q) = a \land \text{Poss}(a, s) \land \text{SR}(a, s) = r \land \delta(q, r) = q' \land s' = \text{do}(a, s)$$

$T^*(q, s, q', s')$ denotes the reflexive transitive closure of $T$, i.e., $T^*(q, s, q', s')$ is true iff starting from program state $q$ and situation $s$, the FSA plan may reach state $q'$ and situation $s'$.
Plan Correctness

\[ T^*(q, s, q', s') \] denotes the reflexive transitive closure of \( T \), i.e., \( T^*(q, s, q', s') \) is true iff starting from program state \( q \) and situation \( s \), the FSA plan may reach state \( q' \) and situation \( s' \), then plan correctness is defined by:

**Definition**

Given a planning problem \( \langle \Sigma, G \rangle \), where \( \Sigma \) is an action theory and \( G \) is a goal formula, a plan axiomatized by \( FSA \) is correct iff

\[
\Sigma \cup FSA \models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s].
\]
Plan Correctness

$T^*(q, s, q', s')$ denotes the reflexive transitive closure of $T$, i.e., $T^*(q, s, q', s')$ is true iff starting from program state $q$ and situation $s$, the FSA plan may reach state $q'$ and situation $s'$, then plan correctness is defined by:

**Definition**

Given a planning problem $\langle \Sigma, G \rangle$, where $\Sigma$ is an action theory and $G$ is a goal formula, a plan axiomatized by $FSA$ is correct iff

$$\Sigma \cup FSA \models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s].$$

Need Second-Order Reasoning!
A planning problem $\langle \Sigma, G \rangle$ is one-dimensional if (intuitively)

- Only one fluent $p$ (called the planning parameter) may take unbounded values from natural numbers;
- All fluents other than $p$ take values from a finite set $V$.
- Initially, $p$ may be arbitrary natural number.
- The only effect on $p$ is to decrease it by one, i.e.,

$$p(\text{do}(a, s)) = x \equiv x = p(s) - 1 \land \text{Dec}(a) \lor x = p(s) \land \neg\text{Dec}(a).$$

- The only primitive test involving $p$ in $\Sigma$ and $G$ is $p = 0$.  


Suppose we are given a one-dimensional planning problem and a candidate FSA plan.
Suppose we are given a one-dimensional planning problem and a candidate FSA plan.

We have verified that the FSA plan correctly achieves the goal for

\[ p = 0, 1, 2, \ldots, N. \]
Suppose we are given a one-dimensional planning problem and a candidate FSA plan.

We have verified that the FSA plan correctly achieves the goal for
\[ p = 0, 1, 2, \ldots, N. \]

Can we now conclude that the FSA plan is correct in general??
The Main Theorem

Theorem

Suppose \( \langle \Sigma, G \rangle \) is a one-dimensional planning problem with planning parameter \( p \), and FSA axiomatize an FSA plan. Then there is an \( N_0 \) such that

\[
\text{If } \Sigma \cup \text{FSA} \cup \{p(S_0) \leq N_0\} \models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s],
\]

then

\[
\Sigma \cup \text{FSA} \models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s].
\]
The Main Theorem

Theorem

Suppose $\langle \Sigma, G \rangle$ is a one-dimensional planning problem with planning parameter $p$, and FSA axiomatize an FSA plan. Then there is an $N_0$ such that

$$\text{If } \Sigma \cup \text{FSA} \cup \{ p(S_0) \leq N_0 \} \models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s],$$

then $\Sigma \cup \text{FSA}$

$$\models \exists s. T^*(Q_0, S_0, Q_F, s) \land G[s].$$
The Main Theorem

Theorem

Suppose \( \langle \Sigma, G \rangle \) is a one-dimensional planning problem with planning parameter \( p \), and FSA axiomatize an FSA plan. Then there is an \( N_0 \) such that

\[
\text{If \ } \Sigma \cup \text{FSA} \cup \{p(S_0) \leq N_0\} \models \exists s. \ T^*(Q_0, S_0, Q_F, s) \land G[s],
\]

\text{then \ } \Sigma \cup \text{FSA} \models \exists s. \ T^*(Q_0, S_0, Q_F, s) \land G[s].

In particular, \( N_0 = 1 + k \cdot m^{|V|} \), where

- \( m \) is the number of finite fluents in \( \Sigma \);
- each such fluent may take at most \( |V| \) different values;
- \( k \) is the number of the program states in the FSA plan.
Proof Sketch

Suppose, for the sake of contradiction, that there is a smallest $N > N_0$ such that if we start from $p = N$ the FSA plan fails.

\[
\langle \vec{b}, N, Q_0 \rangle \rightarrow \langle \vec{b}^{(N_0)}, N, q^{(N_0)} \rangle \rightarrow \\
\langle \vec{b}^{(N_0-1)}, N - 1, q^{(N_0-1)} \rangle \rightarrow \\
\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
\langle \vec{b}^{(2)}, N - N_0 + 2, q^{(2)} \rangle \rightarrow \\
\langle \vec{b}^{(1)}, N - N_0 + 1, q^{(1)} \rangle \rightarrow \text{FAIL} \]

\[
1 + k \cdot m|V| \]

Proof Sketch

Suppose, for the sake of contradiction, that there is a smallest $N > N_0$ such that if we start from $p = N$ the FSA plan fails.

$\langle \vec{b}, N, Q_0 \rangle \rightarrow \langle \vec{b}^{(N_0)}, N, q^{(N_0)} \rangle \rightarrow \langle \vec{b}^{(N_0-1)}, N - 1, q^{(N_0-1)} \rangle \rightarrow \ldots \ldots \langle \vec{b}^*, u, q^* \rangle \rightarrow \ldots \ldots \langle \vec{b}^*, v, q^* \rangle \rightarrow \ldots \ldots \langle \vec{b}^{(2)}, N - N_0 + 2, q^{(2)} \rangle \rightarrow \langle \vec{b}^{(1)}, N - N_0 + 1, q^{(1)} \rangle \rightarrow \text{FAIL}$

$1 + k \cdot m^{\|V\|}$
Proof Sketch

Suppose, for the sake of contradiction, that there is a smallest $N > N_0$ such that if we start from $p = N$ the FSA plan fails.

$$\langle \vec{b}, N, Q_0 \rangle \rightarrow \langle \vec{b}^{(N_0)}, N, q^{(N_0)} \rangle \rightarrow \langle \vec{b}^{(N_0-1)}, N - 1, q^{(N_0-1)} \rangle \rightarrow \ldots \ldots$$

$$\langle \vec{b}^*, u, q^* \rangle \rightarrow \ldots \ldots$$

$$\langle \vec{b}^*, v, q^* \rangle \rightarrow \ldots \ldots$$

$$\langle \vec{b}^{(2)}, N - N_0 + 2, q^{(2)} \rangle \rightarrow \langle \vec{b}^{(1)}, N - N_0 + 1, q^{(1)} \rangle \rightarrow \text{FAIL}$$

Then $\langle \vec{b}, N - (u - v), Q_0 \rangle \rightarrow \text{FAIL}$ too!
Proof Sketch

Suppose, for the sake of contradiction, that there is a smallest $N > N_0$ such that if we start from $p = N$ the FSA plan fails.

\[
\langle \vec{b}, N, Q_0 \rangle \rightarrow \langle \vec{b}^{(N_0)}, N, q^{(N_0)} \rangle \rightarrow \\
\langle \vec{b}^{(N_0 - 1)}, N - 1, q^{(N_0 - 1)} \rangle \rightarrow \\
\hspace{1cm} \ldots \hspace{1cm} \ldots \\
\langle \vec{b}^{\star}, u, q^{\star} \rangle \rightarrow \\
\hspace{1cm} \ldots \hspace{1cm} \ldots \\
\langle \vec{b}^{\star}, v, q^{\star} \rangle \rightarrow \\
\hspace{1cm} \ldots \hspace{1cm} \ldots \\
\langle \vec{b}^{(2)}, N - N_0 + 2, q^{(2)} \rangle \rightarrow \\
\langle \vec{b}^{(1)}, N - N_0 + 1, q^{(1)} \rangle \rightarrow \text{FAIL}
\]

Then $\langle \vec{b}, N - (u - v), Q_0 \rangle \rightarrow \text{FAIL}$ too! \hspace{1cm} \text{Contradiction.}
Towards a Tighter Bound

- $N_0$ is exponential and thus impractical for many cases.
- We proposed an algorithmically obtained bound $N_t$, which is usually much smaller than $N_0$:
  - Verify that the FSA plan is correct for $p = 0, 1, 2, \cdots$.
  - Until for some $N_t$,
    \[
    \langle \mathbf{b}, N_t, Q_0 \rangle \rightarrow \langle \mathbf{b}^*, u, q^* \rangle \rightarrow \langle \mathbf{b}^*, v, q^* \rangle \rightarrow \text{SUCCESS}.
    \]
Experimental Results

We used the different bounds in the test phase of FSAPLANNER on four one-dimensional planning problems (treechop, variegg, safe and logistic).

<table>
<thead>
<tr>
<th>Problem</th>
<th>treechop</th>
<th>variegg</th>
<th>safe</th>
<th>logistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{man}^*$</td>
<td>100</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Time (secs)</td>
<td>0.1</td>
<td>0.12</td>
<td>0.09</td>
<td>3.93</td>
</tr>
<tr>
<td>$N_0$</td>
<td>18</td>
<td>345</td>
<td>4098</td>
<td>514</td>
</tr>
<tr>
<td>Time (secs)</td>
<td>0.03</td>
<td>&gt; 1 day</td>
<td>&gt; 1 day</td>
<td>&gt; 1 day</td>
</tr>
<tr>
<td>$N_t$</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Time (secs)</td>
<td>0.01</td>
<td>0.08</td>
<td>0.08</td>
<td>3.56</td>
</tr>
</tbody>
</table>

*: $N_{man}$ is the manually estimated test bound without correctness guarantee.
Planning with loops is an interesting and challenging problem. In this paper, we
- define a generalized plan representation that allows loops;
- give a formal notion of plan correctness under this representation;
- identify the class of one-dimensional problems whose correctness can be finitely verified;
- show that a planner based on this theoretical result efficiently generates provably correct plans for one-dimensional problems.

Future work: Investigate correctness guarantee for more general classes.
Planning with loops is an interesting and challenging problem. In this paper, we
- define a generalized plan representation that allows loops;
- give a formal notion of plan correctness under this representation;
- identify the class of one-dimensional problems whose correctness can be finitely verified;
- show that a planner based on this theoretical result efficiently generates provably correct plans for one-dimensional problems.

Future work:
- Investigate correctness guarantee for more general classes.
Related Work

- Simple problems for KPLANNER (Levesque 2005);
- Goal achievability for rank 1 theories (Lin 2008);
- Extended-LL problems (Srivastava et al. 2008);
- Abacus programs (Srivastava et al. 2010);
- Deductive approaches (Manna&Waldinger 1987, Magnusson&Doherty 2008);
- Weak guarantee (Winner&Veloso 2007, Bonet et al. 2009);
- Model checking (Clarke et al. 1999).