Planning with Incomplete Knowledge and Limited Quantification

Tamara Babaian
Dept. of Electrical Eng. and Computer Science
Tufts University
Medford, MA 02155 USA
tbabaian@eecs.tufts.edu

James G. Schmolze
Dept. of Electrical Eng. and Computer Science
Tufts University
Medford, MA 02155 USA
schmolze@eecs.tufts.edu
http://www.eecs.tufts.edu/~schmolze

Abstract

We present a new method for partial order planning in the STRIPS/SNLP style. Our contribution centers on how we drop the closed world assumption while adding a useful class of universally quantified propositions to the representation of states and actions. These quantified expressions allow expression of *partially closed worlds*, such as "block A has no other block on it", or "F is the only Tex file in directory D." In addition, we argue informally that the time complexity of our algorithm is no worse than traditional partial order planners that make the closed world assumption.

STRIPS-style planning (Fikes & Nilsson 1971) is decidable only if we restrict the language to finitely many ground terms (Erol, Nau, & Subrahmanian 1992). STRIPS-style planning becomes NP-complete only when we bound the length of the plan being sought (Gupta & Nau 1991). Thus, planning is intractable in the general case. However, thanks to recent advances in applying stochastic search to propositional satisfiability problems (SAT) (Selman, Levesque, & Mitchell 1992) and planning problems (Kautz & Selman 1996), much larger classes of NP-complete planning problems can now be solved. Moreover, recent work demonstrates that studying partial order planning (POP) can lead to good encodings of planning problems for eventual use by stochastic search algorithms (Kautz, McAllester, & Selman 1996).

Our long term research goal is to make planning more realistic by discarding some of the simplifications of the STRIPS-style planning while still remaining in the class of NP complete problems.[1] Specifically, in this paper, we investigate a method for dropping the closed world assumption (CWA) in the context of SNLP-style (McAllester & Rosenblitt 1991) partial order planning (e.g., (Penberthy & Weld 1992; Weld 1994; Russell & Norvig 1995)). A common approach when dropping the CWA retains much of the structure of STRIPS-style planners but re-considers the set of propositions associated with each state of the world. Instead of being a complete description of each state, the set of propositions is considered to be a subset. Thus, with each state we have a set of *beliefs* that are assumed to be true but incomplete (e.g., (Peot & Smith 1992; Etzioni *et al.* 1992; Krebsbach, Olawsky, & Gini 1992; Russell & Norvig 1995)).[2]

The above approach for dropping the CWA is effective but limited in its expressive power if there is no quantification. For example, you can represent a state where the agent knows that directory D has Tex files F and G in it, but you cannot represent that *no other Tex files are in directory D*. To handle this deficit, one can always add special encodings, such as the use of $Clear(x)$ in the blocks world to represent when no other blocks are on x. However, use of $Clear(x)$ is deceptively simple because, in the blocks world, a block can have at most one other block on top. If blocks were allowed to have up to two blocks on top, we would need more complex encodings.

A more direct approach represents the fact that *no other Tex files are in directory D* using a quantified proposition. For example, we might write:

$$In(F,D) \wedge Tex(F) \wedge In(G,D) \wedge Tex(G) \wedge (\forall x \,.\, \neg In(x,D) \vee \neg Tex(x) \vee x = F \vee x = G) \quad (1)$$

Here, $In(x,y)$ is true if and only if (iff) x is currently in directory y and $Tex(x)$ is true iff x is a Tex file. The quantified expression states that every object is either not in D or is not a Tex file *except possibly* F and G – i.e., the quantified expression does not commit to the truth or falsity of $\neg In(F,D) \vee \neg Tex(F)$ nor to $\neg In(G,D) \vee \neg Tex(G)$, but it does commit to $\neg In(x,D) \vee \neg Tex(x)$ for every other x. Using this representation, one can also represent that the truth of specific propositions is unknown. For example,

$$In(F,D) \wedge Tex(F) \wedge In(G,D) \wedge Tex(G) \wedge (\forall x \,.\, \neg In(x,D) \vee \neg Tex(x) \vee x = F \vee x = G \vee x = H)$$

is similar to (1) except that nothing is known about H.

[1] Therefore, in large part, we agree with the strategy proposed by Ginsberg (Ginsberg 1996).

[2] An excellent formal account of this style of representation as applied to planning with sensing actions can be found in (Scherl & Levesque 1997).

We introduce what we call ψ-forms to represent this type of quantified information. While we have studied several classes of ψ-forms, in this paper, we examine only one class that has the following format:

$$\psi \equiv \left\{ \begin{array}{l} \neg Q_1(\vec{x}, \vec{c_1}) \vee \ldots \vee \neg Q_k(\vec{x}, \vec{c_k}) \mid \\ \neg(\vec{x} = \vec{e}_1) \wedge \ldots \wedge \neg(\vec{x} = \vec{e}_n) \end{array} \right\}$$

Here, each Q_i is a predicate symbol, $\vec{x}$ is a vector of variables, each $\vec{c_i}$ is a vector of constants and each $\vec{e_i}$ is a vector of constants. We would thus represent the fact that *no Tex files are in directory D except possibly F and G* with:

$$\{\neg In(x, D) \vee \neg Tex(x) \mid \neg(x = F) \wedge \neg(x = G)\}$$

We prefer to consider (possibly infinite) sets of ground negated clauses instead of the universally quantified sentences from above, although the expressive powers of the two are identical.

In our representation, an agent's knowledge of a given state is a set of propositions, where each proposition is either a ground atom or a ψ-form. We note that a negated literal is a special case of a ψ-form. Throughout this paper, when we refer to negated literals, we actually mean such ψ-forms. We present a POP algorithm for our representation that is a slight modification of the simplest POP algorithm found in (Russell & Norvig 1995). As you will see, we only introduce operations with polynomial time complexity. We thus argue informally that our algorithm is NP-complete if we bound the number of steps and ground terms.

The advantage of ψ-forms is that they can represent useful information and we can reason efficiently with them *as quantified expressions*. We do not need to expand them to a large set of explicit ground literals as is done with certain universally quantified preconditions in UCPOP (Penberthy & Weld 1992). In this paper, our universally quantified ψ-forms can appear in initial state descriptions, goals and in the preconditions of actions. They cannot, however, appear in the effects of actions.

We do not include sensing nor conditional actions nor plan execution in this paper due to space limitations. However, we will present these extensions in a future paper. In that work, ψ-forms can appear in the effects of sensing actions. Our work is similar to the use of *locally closed worlds* (LCWs) in planning (Golden, Etzioni, & Weld 1994; Etzioni, Golden, & Weld 1997), which we examine later.

Our Formalism

We assume that the signature is finite and the only function symbols are constants. Also, any two distinct constants are assumed to denote distinct individuals in all models. Finally, we assume informally that the number of constants is fairly large, at least. If the number of constants is small, then it is probably easier to use ground negated literals in place of quantified ψ-forms.

We base our representation of a changing world on STRIPS (Fikes & Nilsson 1971), which in turn is based on the situation calculus (McCarthy & Hayes 1969). A changing world is thus viewed as a sequence of states, where state transitions occur only as the result of deliberate action taken by the (single) agent.

We use σ to represent a substitution and write $A\sigma$ to represent the formula or term that results from instantiating A by σ. When convenient, we will treat formulas of the form $\vec{A} = \vec{B}$ as substitutions and vice versa. We say that two formulas or terms, A and B, *unify* iff $\exists\sigma \,.\, A\sigma = B\sigma$. We say that A *matches* onto B iff $\exists\sigma \,.\, A\sigma = B$. We write $MGU(A, B)$ to denote the set of most general unifiers (MGUs) of A and B and $MGU_{\subseteq}(A, B)$ to denote the set of MGUs such that $A\sigma \subseteq B\sigma$. Moreover, when we write $MGU(A, B, V)$, we are limiting unification to the variables in the set V. Similarly for $MGU_{\subseteq}(A, B, V)$. Assuming that unification takes constant time (it can be bounded by the size of the largest term), computing $MGU_{\subseteq}(c_1, c_2)$, where c_1 and c_2 are clauses, takes time $O(c^2)$, where c is the maximum number of disjuncts in c_1 and c_2.

Let a *proposition* be either a ground atom or a ψ-form. A ψ-form is a set of negated ground clauses represented as follows.

$$\psi \equiv \{\neg Q_1(\vec{x}, \vec{c_1}) \vee \ldots \vee \neg Q_k(\vec{x}, \vec{c_k}) \mid \neg\sigma_1 \wedge \ldots \wedge \neg\sigma_n\}$$

Here, $Q_i(\vec{x}, \vec{c_i})$ is any atom that uses all and only the variables in $\vec{x}$ and all and only the constants in $\vec{c}$. Each σ_i is just $(\vec{x} = \vec{e_i})$ for some vector of constants $\vec{e_i}$. n is, of course, finite. We also define the following.

- $\mathcal{M}(\psi)$ is the *main part* of ψ, $\neg Q_1(\vec{x}, \vec{c}) \vee \ldots \vee \neg Q_k(\vec{x}, \vec{c})$.
- $\mathcal{V}(\psi)$ denotes the *variables* of ψ, $\vec{x}$, though we usually treat it as a set,
- $\Sigma(\psi)$ is the formula that describes the *exceptions*, $\neg\sigma_1 \wedge \ldots \wedge \neg\sigma_n$.
- $\mathcal{E}_i(\psi)$ is the instantiation of the i-th exception, $(\mathcal{M}(\psi))\sigma_i$. Note that this is *always* ground.
- $\mathcal{E}(\psi)$ is the set of all instantiations of its exceptions, $\{\mathcal{E}_1(\psi), \ldots, \mathcal{E}_n(\psi)\}$.

Note that the cardinality of a ψ-form can be zero.

An *unquantified proposition* is either an atom or a singleton ψ-form. A *negated literal* is a singleton ψ-form whose clause has exactly one term. Moreover, the interpretation of a proposition or of a set of propositions is just the conjunction of all sentences it includes.

A clause can be considered simply as a set of disjuncts. For two clauses c_1 and c_2 we write $c_1 \subseteq c_2$ iff the set of disjuncts of c_1 is a subset of the set of disjuncts of c_2. Therefore, for any two ground clauses c_1 and c_2, $c_1 \models c_2$ iff $c_1 \subseteq c_2$.

Given a set of ground clauses C and a single ground clause c_1, $C \models c_1$ iff $\exists c_2 \in C . c_2 \models c_1$. For any two sets of ground clauses C_1 and C_2 we define *difference*:

$$C_1 - C_2 = \{c \mid c \in C_1 \ \wedge \ C_2 \not\models c\}$$

and *image* (the *image* of C_2 in C_1):

$$C_2 \rhd C_1 = \{c \mid c \in C_1 \wedge C_2 \models c\}.$$

$C_1 - C_2$ is the subset of C_1 that *is not* entailed by C_2 while $C_2 \rhd C_1$ is the subset of C_1 that *is* entailed by C_2. Thus, $(C_1 - C_2)$ and $(C_2 \rhd C_1)$ always partition C_1. Moreover, we have the following equivalences.

1. $C_1 - C_2 = C_1 - (C_2 \rhd C_1)$ and
2. $C_2 \rhd C_1 = C_1 - (C_1 - C_2)$

The image operation is critical, as we will show, because a clause may be entailed by several different other clauses, and vice versa.

Let a *world state*, W, be a consistent set of propositions where every atom A is either true or false. Thus, a world state is complete and has only definite knowledge. Let a *state of knowledge* (SOK) be a consistent set of propositions that represents the knowledge that our (single) agent has about a particular world state. Our agent, and thus our planner, never has access to complete world states, but only to SOKs. We assume throughout this paper that an agent's SOK is always correct.

An *action* is ground and is represented in a fashion similar to STRIPS. However, we do not use the add-list and delete-list format because it would be possible to write action descriptions that produce inconsistent states. Instead, each action a has:

- a *name*, $\mathcal{N}(a)$,
- a set of propositions called the *preconditions*, $\mathcal{P}(a)$ (which may include general ψ-forms), and
- a set of literals called the *assert list*, $\mathcal{A}(a)$ (i.e., all ψ-forms here must be negated literals).

The precondition identifies the conditions necessary for executing the action. The assert list, also called the *effects* of the action, identifies *all and only* the propositions that change as a result of the action.

It is possible to execute an action a in a world state W iff $W \models \mathcal{P}(a)$. If executed from world state W, the state that results is consistent with $(W - \mathcal{A}(a)^\circ) \cup \mathcal{A}(a)$.[3] Here, we define S° be the *closure* of S. For either an atom A or a negated literal $\neg A$, the closure is the set $\{A, \neg A\}$. For a set of propositions, the closure is the union of the closures of its members. We first remove the closure of the assert list so that we can add the assert list back without conflict.

However, our agent will only have SOKs. Thus, it will only execute an action a iff its SOK about the current state is S and $S \models \mathcal{P}(a)$. The agent's SOK about the state that results is

$$S' = \left(S - \mathcal{A}(a)^\circ - \bigcup_{\forall p \in \mathcal{A}(a)} \bigcup_{\forall \psi \in S} (\{\neg p\} \rhd \psi) \right) \cup \mathcal{A}(a).$$

For an example, we characterize the action $a = mv(fig, /img, /tex)$, which moves the file *fig* from directory */img* into */tex*. We use $PS(x)$ to represent that file x is in postscript format. Let $\mathcal{P}(a) = \{In(fig, /img)\}$ which states that *fig* must be in */img*. Also, let $\mathcal{A}(a) = \{\neg In(fig, /img), In(fig, /tex)\}$. We begin with an SOK:

$$S = \left\{ \begin{array}{l} In(a.tex, /tex), In(fig, /img), PS(a.ps), \\ \{\neg In(x, /img) \mid \neg(x = fig)\}, \\ \{\neg In(x, /tex) \vee \neg PS(x) \mid \neg(x = a.ps)\} \end{array} \right\}$$

$a = mv(fig, /img, /tex)$ is executable, and the resulting SOK is:

$$S' = \left\{ \begin{array}{l} In(fig, /tex), In(a.tex, /tex), PS(a.ps), \\ \{\neg In(x, /img)\}, \\ \{\neg In(x, /tex) \vee \neg PS(x) \mid \\ \quad \neg(x = a.ps) \wedge \neg(x = fig)\} \end{array} \right\}$$

Note that S contained $\neg In(fig, /tex) \vee \neg PS(fig)$ and that we added $In(fig, /tex)$ when determining S'. If our update rule retained $\neg In(fig, /tex) \vee \neg PS(fig)$ in S', then in S' we could perform resolution and conclude that $\neg PS(fig)$. However, this would be wrong because we have no information on *fig* being a postscript file or not. Instead, our update rule deletes any clause that is entailed by $\neg In(fig, /tex)$, and so S' does not contain $\neg In(fig, /tex) \vee \neg PS(fig)$.

While preconditions can use arbitrary ψ-forms, the assert list can only use literals. The reasons for this limitation is that we are only interested in simple[4] actions (for now), and that each action must identify *exactly* those propositions that change due to the action. This precision is needed so that we can accurately compute the state that results from the action. Thus, removing an object from a briefcase, or copying a file, are actions that fit into our model while the action of removing all files from a directory do not.[5]

There is, however, one notable exception. We allow quantified ψ-forms in the START action, whose assert list holds the initial SOK for a planning problem, which we assume is consistent. Thus, in this paper, the initial SOK is the only place where ψ-forms can be introduced into a problem.

For this initial SOK, we compute all possible resolutions and add them back to the SOK. The number of such resolutions is limited by the number of atoms in the SOK. Thus, the initial SOK is *saturated* in the sense that any atom entailed by the SOK is explicitly in the SOK and for any negated clause entailed, there

[3] Note that set difference operator, "–", requires further definition, which we delay until the next section. However, its meaning in this context is the obvious one.

[4] By simple we mean an action that has causal (Etzioni *et al.* 1992) effects only and makes only finite number of changes to the state. We also assume it is deterministic. All effects of a simple action can therefore be described by a finite set of literals, without universal quantification in the effects.

[5] For actions like *rm**, we cannot use ψ-forms to describe the effects because we require knowing precisely which propositions changed. ψ-forms could be used with *rm** to describe the state of affairs that results (i.e., all files in directory are removed), but this is not helpful because they cannot identify precisely which files were removed.

is a sub-clause in the SOK. As it turns out, all SOKs after the initial SOK are consistent and similarly saturated thanks to our update rule above.

Entailment and ψ-difference

We now present both theory and methods for determining entailment and set difference for sets of propositions that may include ψ-forms. Essentially, we translate these tests into operations of unification, instantiation, identity, and simple set operations. The methods presented here via theorems are readily translated into algorithms. Due to space limits, we do not present proofs in this paper.

Entailment

Everywhere in this section $A, A_1, \ldots, A_n$ denote atoms and $\psi, \psi_1, \ldots, \psi_n$ denote ψ-forms. Also, C and E denote maximum number of disjuncts and exceptions in a ψ-form respectively, L is the maximum number of ψ-forms in a SOK, and N is the maximum number of literals in any action's assert list.

Let a *simple* ψ-form be one that has no exceptions – i.e., $\mathcal{E}(\psi) = \emptyset$ (and thus, $\Sigma(\psi) = true$).

Theorem 1 *Let $\psi, \psi_1, \ldots, \psi_n$ be simple ψ-forms. $\{\psi_1, \ldots, \psi_n\} \models \psi$ iff*

$$\exists i . (1 \leq i \leq n) \wedge (\psi_i \models \psi).$$

Thus, if we ignore exceptions, then to show that a set of ψ-forms entails ψ, we need find only one element of the set that entails ψ. This turns out to be a critical factor in keeping ψ-form reasoning tractable as there is no need to examine combinations of ψ-forms for this.

Theorem 2 *Let ψ and ψ' be simple ψ-forms where variables are renamed as needed so there is no variable overlap between ψ and ψ'. $\psi' \models \psi$ iff $\exists \sigma . (\mathcal{M}(\psi')\sigma \subseteq \mathcal{M}(\psi))$ or, equivalently, $MGU_{\subseteq}(\mathcal{M}(\psi'), \mathcal{M}(\psi), \mathcal{V}(\psi')) \neq \emptyset$.*

Thus, to test whether a simple ψ-form entails another, we simply test whether there is a subset-match between the main part of the former and the main part of the latter. The time complexity for this test is $O(C^2)$.

To determine whether or not $\{\psi_1, \ldots, \psi_n\} \models \psi$ in general,we first find a ψ_i whose main part entails the main part of ψ. However, the exceptions of ψ_i weakens ψ_i. Therefore, we must also account for every clause in ψ not implied by ψ_i, i.e. $\psi - \psi_i$ which is exactly the set $(\mathcal{E}(\psi_i) \rhd \psi) - (\psi_i \rhd \psi)$ (see Theorem 7). The following two theorems describe the procedure for calculating this set, and Theorem 5 defines necessary and sufficient conditions for ψ-form entailment.

Theorem 3 *For an arbitrary finite set of ground clauses $\{c_1, \ldots, c_n\}$*

$$\{c_1, \ldots, c_n\} \rhd \psi = \{\mathcal{M}(\psi)\sigma \mid \sigma \in \Omega\} - \mathcal{E}(\psi)$$

where $\Omega = \bigcup_{j=1}^{n} MGU_{\subseteq}(c_j, \mathcal{M}(\psi))$, and is a set of size $O(nC)$.

Thus, computing $\{c_1, \ldots, c_n\} \rhd \psi$ takes time $O(nC(C+E))$.[6]

Theorem 4 *For an arbitrary finite set of ground clauses $\{c_1, \ldots, c_n\}$*

$$\begin{aligned}&(\{c_1, \ldots, c_n\} \rhd \psi_p) - ((\psi - \{c_1, \ldots, c_n\}) \rhd \psi_p) = \\ &\{c_p \mid c_p \in (\{c_1, \ldots, c_n\} \rhd \psi_p) \wedge \\ &\quad \forall \sigma \in MGU_{\subseteq}(\mathcal{M}(\psi), c_p).[\mathcal{M}(\psi)\sigma \in \mathcal{E}(\psi) \vee \\ &\quad \mathcal{M}(\psi)\sigma \in \{c_1, \ldots, c_n\}]\}\end{aligned}$$

is a set of size $O(nC)$.

Theorem 5 *$\{\psi_1, \ldots, \psi_n \models \psi\}$ iff*

$$\begin{aligned}&\exists i.(1 \leq i \leq n) \wedge (\{\mathcal{M}(\psi_i)\} \models \{\mathcal{M}(\psi)\}) \wedge \\ &\forall c \in (\psi - \psi_i).\exists k.(1 \leq k \leq n) \wedge \psi_k \models c\end{aligned}$$

Difference Operation Among ψ-forms

Our planner encounters only two special cases of $\psi_2 - \psi_1$, namely when ψ_1 is a singleton ground clause and also when the main part of ψ_1 entails the main part of ψ_2.

When ψ_1 is a singleton ground clause and ψ_1 does not entail any clause in ψ_2, then $\psi_2 - \psi_1 = \psi_2$. Otherwise, $(\psi_1 \rhd \psi_2)$ become new exceptions in the difference.

Theorem 6 *Let ψ_1 be a singleton ground clause . If $(\psi_1 \rhd \psi_2) = \emptyset$ then $\psi_2 - \psi_1 = \psi_2$. Otherwise,*

$$\psi_2 - \psi_1 = \{\, \mathcal{M}(\psi_2) \mid \Sigma(\psi_2) \wedge \neg\sigma_1 \ldots \wedge \neg\sigma_k \,\}$$

where $\{\sigma_i\}_{i=1}^{k} = MGU_{\subseteq}(\psi_1, \mathcal{M}(\psi_2))$.

We know that the above $\{\sigma_i\}_{i=1}^{k} \neq \emptyset$ when $(\psi_1 \rhd \psi_2) \neq \emptyset$. The time complexity of this calculation is the same as of computing $(\{c\} \rhd \psi_2)$, namely, O(C(C+E)).

When $\{\mathcal{M}(\psi_1)\} \models \{\mathcal{M}(\psi_2)\}$, the only residuals from $\psi_2 - \psi_1$ are the clauses in $(\mathcal{E}(\psi_1) \rhd \psi_2)$ that are not implied by any clause of ψ_1.

Theorem 7 *Let $\{\mathcal{M}(\psi_1)\} \models \{\mathcal{M}(\psi_2)\}$. Then*

$$\psi_2 - \psi_1 = (\mathcal{E}(\psi_1) \rhd \psi_2) - (\psi_1 \rhd \psi_2)$$

As follows from Theorem 4, $\psi_2 - \psi_1$ is a finite set of singleton ground clauses , i.e. set of ψ-forms, and the time complexity for computing $\psi_2 - \psi_1$ is $O(EC^2(E+C))$.

Combining the complexity measures for the above computation with the result of Theorem 5, we conclude that ψ-form entailment in a SOK takes $O(EC^2L(C+E))$ in the worst case. Assuming all propositions in SOK are stored in a hash table, time complexity of the state update rule is $O(NLC^2(C+E))$.

[6] In comparison to LCW work (Golden, Etzioni, & Weld 1994; Etzioni, Golden, & Weld 1997) our E term roughly corresponds to M - the number of all ground literals in their main database.

Algorithm. POP ($<S, O, L>$, *open*)

1. If *open* is empty, return $<S, O, L>$
2. Pick a goal $< c, S_c >$ from *open* and remove it from *open*. **choose** an existing step S_s from S, or a new step S_s, that has an effect e where $e \models c$. If no such step exists then **fail**.
3. Add link $S_s \xrightarrow{e,c} S_c$ to L.
4. Add $S_s \prec S_c$ to O.
5. if S_s is a new step:
   - Add START$\prec S_s$ and $S_s \prec$FINISH to L.
   - For each p in $\mathcal{P}(S_s)$ (the preconditions of S_s), add $< p, S_s >$ to *open*.
6. For every step S_t that threatens a link $S_s \xrightarrow{e,p} S_c$ nondeterministically **choose** either:
   - *Demotion:* Add $S_t \prec S_s$ to O.
   - *Promotion:* Add $S_c \prec S_t$ to O.
7. If O is inconsistent then **fail**.
8. Recursively call POP with updated $<S, O, L>$ and *open*.

Figure 1: Original POP algorithm. Triple $<S, O, L>$ denotes a partial plan; S is a set of *steps*, which are (ground) actions, initially contains only START and FINISH; O is a set of *ordering constraints* of the form $S_i \prec S_j$, where S_i and S_j are steps in S, initially contains START$\prec$FINISH; L is a set of (causal) *links* of form $S_i \xrightarrow{e,p} S_j$, where p is a precondition of S_j, e is an effect of S_i (i.e., e is in the assert list of S_i), and $e \models p$. We call S_i and e the *source* step and proposition, and S_j and p the *target* step and proposition. L is initially empty. *open* is the list of open preconditions; initially contains preconditions of the FINISH step.

POP Algorithm with ψ-forms

We first present the algorithm for partial order planning (POP), which we take largely from (Russell & Norvig 1995), and then describe modifications needed for including ψ-forms. We assume that the reader is already familiar with SNLP-style planning (McAllester & Rosenblitt 1991), and will rely upon the tests that are defined in the previous section.

Figure 1 shows the POP algorithm written for a nondeterministic machine. We made two minor changes to the standard algorithm so that it easily generalizes to handling ψ-forms. First, our links have both source and target conditions, which may differ – a link in standard POP has only one condition since the conditions on the source and target steps must be identical. Second, in step 2, we pick a step whose effect $e \models c$ – in standard POP, we pick a step whose effect is exactly the same as c.

Several changes are needed for the POP algorithm to handle ψ-forms. First we preprocess the initial SOK and perform all possible resolutions. The other modifications affect steps 2 and 6.

Modifying Step 2

In step 2, we nondeterministically seek every step S_s that has an effect e where $e \models c$. It should be clear that when c is an atom, then e must be an atom, and when c is a ψ-form, then so must e. In the latter case, however, we also seek steps where e does *not* entail c but where e does meet the following criterion: the main part of e entails the main part of c. Then we perform *goal splitting*.

If we have two ψ-forms, ψ_c and ψ_e, where both are not single clauses and where $\{\mathcal{M}(\psi_e)\} \models \{\mathcal{M}(\psi_c)\}$, then "most" of ψ_c is entailed by ψ_e. The only "leftovers" are the clauses of ψ_c for which there is no entailment from any clause of ψ_e. These clauses are a subset of clauses implied by exceptions of ψ_e. In fact, this set of "leftovers" is precisely $\psi_c - \psi_e$ as defined in Theorem 7. The result of this difference is a set of singleton ground clauses – i.e., unquantified ψ-forms. Thus, *goal splitting* is the act of taking such a ψ_c and ψ_e and splitting ψ_c into a set of ψ-forms, $\Psi = \{\psi\} \cup (\psi_c - \psi_e)$, where $\psi = \psi_e \rhd \psi_c = \psi_c - (\psi_c - \psi_e)$. Here, ψ is just ψ_c after we add more exceptions to it as defined by Theorem 6. Then, we remove ψ_c from step S_c and replace it with the equivalent set of ψ-forms in Ψ. Next, we add a link from ψ_e to ψ, namely $S_s \xrightarrow{\psi_e,\psi} S_c$. Finally, we add the new ψ-forms to *open*. In this way, we have split ψ_c into a quantified ψ-form, ψ, which is linked from ψ_e, plus a set of singleton ground clause ψ-forms, namely $\psi_c - \psi_e$, that still need links.

Example 1. Let $\psi_e = \{\neg P(x, y) \mid \neg(xy = ab) \wedge \neg(xy = aa)\}, \psi_c = \{\neg P(z, b) \vee \neg P(a, z)\}$. Then $\{\mathcal{M}(\psi_e)\} \models \{\mathcal{M}(\psi_c)\}$. $\mathcal{E}(\psi_e) \rhd \psi_c = \{\neg P(a, b) \vee \neg P(a, a), \neg P(b, b) \vee \neg P(a, b)\}$. The second clause in this set is entailed by ψ_e's clause $\neg P(b, b)$, and therefore $\psi_c - \psi_e = \{\neg P(a, b) \vee \neg P(a, a)\}$. Thus after the goal splitting ψ_c is set to $\{\neg P(z, b) \vee \neg P(a, z) \mid \neg(z = a)\}$, causal link $S_s \xrightarrow{\psi_e,\psi_c} S_c$ is created, and a new subgoal $\{\neg P(a, b) \vee \neg P(a, a)\}$ is added to S_c and *open*.

Note that goal splitting is an equivalence-preserving transformation.

Modifying Step 6

For a step to be a threat, its effect must remove the supporting proposition for the precondition from the SOK , ψ-forms can only be threatened by atoms, and vice versa.

We add goal splitting to the arsenal of threat resolution techniques. Goal splitting applies when an atom threatens a causal link that supports a quantified ψ-form. An atom c that is an effect for step S_s is a threat to a causal link $S_e \xrightarrow{\psi_e,\psi_p} S_p$ iff it removes all clauses of ψ_e that support some clause(s) in ψ_p. In particular,

the clauses of ψ_p that lose support as the result of the threat, c, are exactly

$$\psi \equiv ((\{\neg c\} \rhd \psi_e) \rhd \psi_p) - ((\psi_e - \{\neg c\}) \rhd \psi_p)$$

Here, $(\{\neg c\} \rhd \psi_e)$ are those clauses that will be "lost" from ψ_e due to effect c of S_s, and $(\{\neg c\} \rhd \psi_e) \rhd \psi_p$ are those clauses in ψ_p that depend on those "lost" supports. However, some of these clauses in ψ_p might also be entailed by other clauses in ψ_e that are not going to be "lost" due to step S_s. The clauses that will not be "lost" from ψ_e are $(\psi_e - \{\neg c\})$ and the clauses in ψ_p that benefit from these remaining clauses are $((\psi_e - \{\neg c\}) \rhd \psi_p)$. Note that ψ is empty if $\{\neg c\} \rhd \psi_e = \emptyset$.

Applying goal splitting, we first find all such clauses, remove them from ψ_p (as defined in Theorem 6) and add them to the preconditions of S_p and to *open*. The link $S_e \overset{\psi_e,\psi_p}{\rightarrow} S_p$ is still valid, because removing the threatened clauses from ψ_p weakens it so that c is no longer a threat to $S_e \overset{\psi_e,\psi_p}{\rightarrow} S_p$. Thus, the threat is resolved. However, we must find new links to support the clauses in ψ.

Example 2. Let l be the link $S_s \overset{\psi_e,\psi_c}{\rightarrow} S_c$ defined in the previous example. Effect $c_1 = P(c,d)$ is clearly not a threat to l, because $((\{\neg c_1\} \rhd \psi_e) \rhd \psi_c) = \emptyset$.

Let's check if $c_2 = P(c,b)$ is a threat to l. $((\{\neg c_2\} \rhd \psi_e) \rhd \psi_c) = \{\neg P(c,b) \vee \neg P(a,c)\}$, which is also implied by ψ_e's clause $\neg P(a,c)$. Thus, $((\{\neg c_2\} \rhd \psi_e) \rhd \psi_c) \subseteq ((\psi_e - (\{\neg c_2\} \rhd \psi_e)) \rhd \psi_c)$ and c_2 is not a threat to l.

Example 3. Let l be the link $S_s \overset{\psi_e,\psi_c}{\rightarrow} S_c$, where $\psi_e = \{\neg P(x) \vee \neg Q(x)\}, \psi_c = \{\neg P(y) \vee \neg Q(y) \vee \neg R(y)\}$, and $c = P(a)$. c is a threat to l, because it removes support from ψ_c's clause $\neg P(a) \vee \neg Q(a) \vee \neg R(a)$. The result of goal splitting is $\psi_c = \{\neg P(y) \vee \neg Q(y) \vee \neg R(y) \mid \neg(y = a)\}$, and a new precondition $\neg P(a) \vee \neg Q(a) \vee \neg R(a)$ in S_c and *open*.

Note that a situation in which a non-singleton ψ-form threatens a causal link supporting an atom goal is not possible, simply because non-singleton ψ-form effects appear only in the START step, and every other step in the plan is ordered after START. Singleton ψ-forms, however, are handled by the original POP algorithm, and so we do not need any changes in case the threatened link supports an atom goal.

This algorithm, which we believe (but have not shown) to be sound and complete when the domain of objects is infinite, can be extended to handle conditional effects and work with non-ground actions.

Comparison with LCWs

As mentioned above, there are numerous works on STRIPS-style planning without the closed world assumption. To our knowledge, however, only the LCW work (Golden, Etzioni, & Weld 1994; Etzioni, Golden, & Weld 1997) deals with the type of quantified information that is similar to our ψ-forms. The planner presented in these papers handles sensing actions and information loss, which we do not address here.

An LCW sentence represents local closed world information. LCW(Φ) means that for all ground substitutions θ, the agent knows the truth value of $\Phi\theta$. Syntax of Φ sentences is limited to the conjunction of positive literals (which corresponds exactly to the disjunction of negative literals in simple ψ-forms) and cannot express statements with exceptions, like "we know sizes of all files, except *a.tex* and *a.ps*", which can be expressed with a ψ-form.

Because of this deficiency, the changes in the world state cannot be accurately reflected in the agent's knowledge database consisting of ground literals and LCW sentences, and as a result some information gets discarded. The LCW reasoning in itself is incomplete even without sensing and information loss. The query mechanism is also incomplete - it cannot deduce all ground facts that are implied by the agent's database.

The language of ψ-forms doesn't have any of these problems and thus is more adequate in representing incomplete knowledge. It allows us to express statements with exceptions, while keeping the reasoning complete and tractable. Time complexity measures of state update rule and ψ-form entailment, for example, compare favorably to those of LCW language.

The ψ-forms presented in this paper have a limitation that every disjunct in a ψ-form must use all the same variables, and thus there's a "gap" between the expressive power of simple ψ-forms and LCW sentences which have no such restriction on their conjuncts. For example, the constraint $\{\neg PS(x) \vee \neg In(x,y)\}$ cannot be represented with a ψ-form used in the presented language, because its first disjunct uses only one of the two quantified variables, but can be encoded as an LCW sentence. However, if we drop that requirement and also allow for non-ground exceptions, the ψ-form language becomes strictly more expressive than LCW, while retaining completeness and tractability of reasoning.

Finally, ψ-forms can be used efficiently to describe the effects of sensing actions.

Conclusions and Future Work

We have presented a method for SNLP-style partial order planning (POP) that does not make the closed world assumption and that allows limited quantification. The key idea is the use of ψ-forms to represent quantified negative information and to integrate ψ-forms into POP such that it adds only polynomial cost algorithms. We thus argue informally that, even with our expanded representation, we can keep the complexity of planning within NP if we have a finite language and if we bound the length of plans. The extension of the presented algorithm to a lifted version with conditional effects is straightforward.

Future work is already in progress. We will continue to explore richer representations for ψ-forms, particu-

larly the use of non-ground exceptions. Also, we will continue developing methods for integrating sensing, conditional planning and plan execution. Finally, we will formally examine the issues of soundness, completeness and complexity of the planning algorithm.

References

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1992. When is Planning Decidable? In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS-92)*, 222–227.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An Approach to Planning with Incomplete Information. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, 115–125.

Etzioni, O.; Golden, K.; and Weld, D. 1997. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence* 89(1–2):113–148.

Fikes, R., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Ginsberg, M. L. 1996. Do Computers Need Common Sense? In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, 620–626.

Golden, K.; Etzioni, O.; and Weld, D. 1994. Omnipotence Without Omniscience: Efficient Sensor Management for Planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1048–1054. Seattle, WA: American Association for Artificial Intelligence.

Gupta, N., and Nau, D. S. 1991. Complexity Results for Blocks-World Planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 629–633. Anaheim, CA: American Association for Artificial Intelligence.

Kautz, H., and Selman, B. 1996. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1194–1201. Portland, OR: American Association for Artificial Intelligence.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding Plans in Propositional Logic. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, 374–384.

Krebsbach, K.; Olawsky, D.; and Gini, M. 1992. An Empiracal Study of Sensing and Defaulting in Planning. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS-92)*, 136–144.

McAllester, D., and Rosenblitt, D. 1991. Systematic Nonlinear Planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 634–639.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine intelligence 4*. New York: American Elsevier.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, 103–114.

Peot, M. A., and Smith, D. E. 1992. Conditional Nonlinear Planning. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS-92)*, 189–197.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

Scherl, R. B., and Levesque, H. J. 1997. The Frame Problem and Knowledge-Producing Actions. Submitted to *Artificial Intelligence*..

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 440–446. San Jose, CA: American Association for Artificial Intelligence.

Weld, D. S. 1994. An Introduction to Least Commitment Planning. *AI Magazine* 15(4):27–61.