UCPOP: A Sound, Complete, Partial Order Planner for ADL

J. Scott Penberthy
IBM T.J. Watson Research Center
P.O Box 704
Yorktown Heights, NY 10598
jsp@watson.ibm.com

Daniel S. Weld
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98105
weld@cs.washington.edu

Abstract

We describe the UCPOP partial order planning algorithm which handles a subset of Pednault's ADL action representation. In particular, UCPOP operates with actions that have conditional effects, universally quantified preconditions and effects, and with universally quantified goals. We prove UCPOP is both sound and complete for this representation and describe a practical implementation that succeeds on all of Pednault's and McDermott's examples, including the infamous "Yale Stacking Problem" [McDermott 1991].

1 Introduction

The investigation of techniques for reasoning about actions and plans is split into two camps. One camp has looked at formal characterizations of languages for describing change while another has attempted to build actual planners, often losing a precise understanding of their programs in a forest of pragmatic choices. A few researchers have described complete algorithms rigorously, but all these approaches suffer from one of two liabilities.

- Either the planners only handle the restrictive STRIPS representation (*e.g.,* TWEAK [Chapman 1987] and SNLP [McAllester and Rosenblitt 1991]),
- or the planners represent plans as totally ordered sequences of actions (*e.g.,* Rosenschein's [1981] and Kautz's [1982] bigression planners, and McDermott's [1991] PEDESTAL).

Since consensus suggests that partial order planning is preferable to total order approaches [Minton *et al.* 1991,Barrett *et al.* 1991,Barrett and Weld 1992], we pondered the void in the space of rigorous planners. McDermott [1991] clearly believed our quest doomed: "if you want a completeness theorem for your planner, you had better build a linear planner."

In this paper we show McDermott was overly pessimistic. We describe UCPOP, a Partial Order Planner whose step descriptions include Conditional effects and Universal quantification.[1] Both universal and existential quantification are permitted in the step preconditions, effect preconditions, effect postconditions, and goals. Although UCPOP assumes a closed world (actions cannot add or delete from the fixed universe of objects) and does not allow domain axioms or disjunctive preconditions, it is considerably more expressive than other rigorous partial order planners.

The UCPOP algorithm starts with an initial, dummy plan that consists solely of a "start" step (whose effects encode the initial conditions) and a "goal" step (whose preconditions encode the goals). UCPOP then attempts to complete this initial plan by adding new steps and constraints until all preconditions are guaranteed to be satisfied. The main loop makes two types of choices: supporting "open" preconditions and resolving "threats."

- If UCPOP has not yet satisfied a precondition (*i.e.,* it is "open"), then all step effects that could possibly be constrained to unify with the desired proposition are considered. UCPOP chooses one effect nondeterministically[2] and then adds a causal link [McAllester and Rosenblitt 1991] to the plan to record this choice.
- If a third step (called a "threat") might possibly interfere with the precondition being supported by the causal link, then UCPOP nondeterministically chooses a method to resolve the threat: either by reordering steps in the plan, posting additional subgoals, or by adding new equality constraints.

[1] The name, which we pronounce as "yoo-see-pop", is an anagram of the capitalized letters.

[2] In fact, domain dependent information can be used to guide the choice. Backtracking ensures that all choices will be eventually considered.

Once UCPOP has successfully created and protected a causal link for every goal in the plan, it halts and returns a solution.

1.1 Money at Home, Wisdom at Work

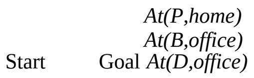
Adopting an example from [Pednault 1988], suppose we had a single briefcase, B, and wanted to use it to move objects. Pednault formalizes this simple domain with three operators: `MovB(l)`, for moving a briefcase and its contents, `PutIn(x)` for putting an item x in the briefcase, and `TakeOut(x)` for removing items from the briefcase. Our version of these operators is seen below:

`TakeOut(x)`
  PRECOND : $\mathtt{x} \neq \mathtt{B}$
  EFFECTS : $\neg\mathtt{In(x)}$

`PutIn(x,l)`
  PRECOND : $\mathtt{x} \neq \mathtt{B} \wedge \mathtt{At(B,l)} \wedge \mathtt{At(x,l)}$
  EFFECTS : `In(x)`

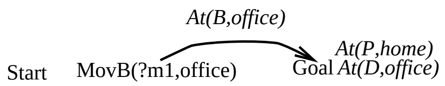
`MovB(m,l)`
  PRECOND : $\mathtt{At(B,m)} \wedge \mathtt{m} \neq \mathtt{l}$
  EFFECTS : `At(B,l)`
  $\mathtt{At(z,l)}\ \forall \mathtt{z} \mid \mathtt{In(z)} \wedge \mathtt{z} \neq \mathtt{B}$
  $\neg\mathtt{At(B,m)}$
  $\neg\mathtt{At(z,m)}\ \forall \mathtt{z} \mid \mathtt{In(z)} \wedge \mathtt{z} \neq \mathtt{B}$

We now demonstrate how UCPOP generates plans with these actions. Since the algorithm is nondeterministic, we assume a convenient order; in practice, some backtracking might be required to find a correct plan. Our example begins with three items: a briefcase B, a paycheck P, and a dictionary D. All three start at home, with the briefcase containing the paycheck. The goal is to have the paycheck at home and the dictionary at the office. The initial plan is depicted as follows:



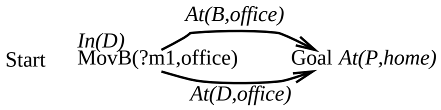
In each diagram we use *italics* to represent open preconditions which are treated as subgoals. When these preconditions are still open (*i.e.*, have not been satisfied), we display them next to the step that requires them. Steps, which are instances of operators, are written in `typeface`. Arrows between steps denote causal links, showing which subgoals a step has satisfied.

UCPOP selects the goal *At(B,office)*, satisfying it by creating a new `MovB(?m1,office)` step and making a causal link from it to the `Goal` step.[3]

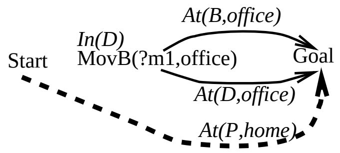


[3] Although this creates another subgoal *At(B,?m1)* we have omitted it and its links for simplicity.

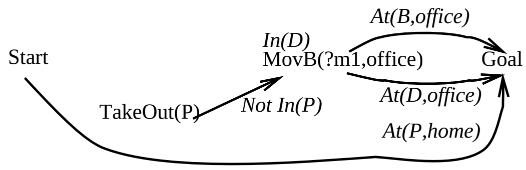
UCPOP then selects the subgoal *At(D,office)*, creating another link from the same step used in the previous goal. This adds a subgoal *In(D)* to the `MovB(?m1,office)` step:



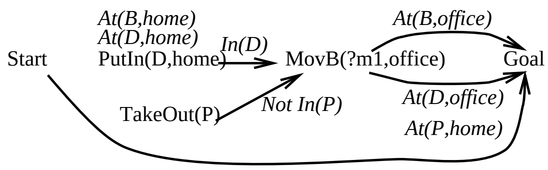
The next goal selected is *At(P,home)*. Since the paycheck is at home to begin with, UCPOP can use this fact to satisfy the final goal as well; UCPOP records this decision by adding a link from `Start` as shown:



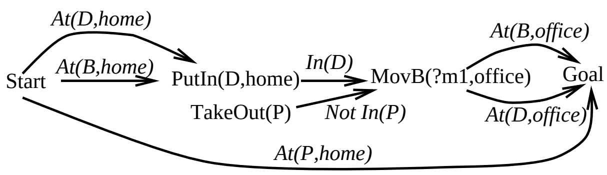
The dashed link denotes a *threat* from the `MovB` step: if the paycheck is left in the briefcase, then moving the briefcase will negate the initial condition, *At(P,home)*, and jeopardize the supported goal. UCPOP eliminates the threat by posting a subgoal $\neg In(P)$ which, in turn, is satisfied by adding `Takeout(P)` to the plan:



UCPOP then selects the subgoal *In(D)* and creates a causal link from another new step `PutIn(D,?m1)`, ensuring that the dictionary is in the briefcase before the briefcase is moved. The new step also generates two more subgoals to ensure that the briefcase and dictionary are spatially coincident:



Finally, UCPOP generates two more links from the initial `Start` step to satisfy the preconditions of the `PutIn` step. This also causes the free variable $?m1$ to become bound to *home*. Since no more unsatisfied subgoals are found and since no threats are detected, UCPOP halts with the following partially ordered plan:



	Run Time	Unifies	Plans
Briefcase	510	177	12
U Briefcase	670	160	11
Homeowners	580	48	24
Sussman	690	233	12
U Sussman	1810	1065	20
YSP	2410	1768	20

Figure 1: These data are taken from an implementation of UCPOP on an IBM RS/6000 model 520. Times are in milliseconds.

1.2 Pragmatics and Implementation

The UCPOP algorithm has been implemented in Allegro Common Lisp and runs on a variety of platforms.[4] To maintain completeness we use A* or IDA* search to implement the exploration of nondeterministic choices. Figure 1 shows the actual run times for the briefcase example and others found in the literature [McDermott 1991,Pednault 1986,Pednault 1988,Sussman 1975]. The number of unifications and the total number of partial plans that UCPOP considered when solving a particular planning problem are also shown.

The problems in figure 1 are summarized as follows:

- **Briefcase.** Pednault's conditionalized briefcase domain. The "U" before a problem indicates the use of universal quantification in postconditions, as described in this paper.
- **Homeowners.** You bought a home, turn on the water, and water pours out of holes in the wall. You can paste over the holes and/or fix the plumbing. However, pasting holes before fixing the plumbing is useless [Pednault 1991].
- **Sussman.** We tested on two versions of the Sussman anomaly. The "U Sussman" problem foregoes the $clear(b)$ axiom and instead uses $\forall x\ \neg on(x, b)$, introducing numerous subgoals.
- **YSP.** McDermott's *Yale Stacking Problem*, a variation of the Sussman anomaly that uses the *above* predicate.

We observe that UCPOP found the Yale Stacking Problem quite easy, repudiating McDermott's [1991] belief that it would be "almost impossible for a nonlinear planner. (p 405)" The combinatorics of handling universal preconditions and multiple interacting steps are exemplified by increasing time.

1.3 Overview

In this paper we make the following contributions:

[4] Send mail to `weld@cs.washington.edu` for information on acquiring the code.

- A clear and simple description of the UCPOP planning algorithm.
- A proof of UCPOP's soundness.
- A proof of UCPOP's completeness.

In the next section we review Pednault's ADL (the language UCPOP uses to represent actions) and detail a few simplifying assumptions. We then (sections 3 and 4) describe a representation for partially ordered plans and present the UCPOP algorithm. Section 5 presents our core results: proofs of soundness and completeness. The paper concludes with a brief discussion of related work.

2 Representing Actions

Frustrated with the restrictive STRIPS representation but frightened by the prospect of implementing a planner for the full situation calculus, we gravitated naturally toward Pednault's [1986,1989] Action Description Language (ADL). ADL is essentially a reformulation of the situation calculus into *action schemata*, akin to the add and delete lists of STRIPS [Fikes and Nilsson 1971]. ADL is more expressive than STRIPS yet less expressive than full, first-order logic.

2.1 Action schemata

The semantics of ADL are based on the algebraic structures (models) used to characterize states of the world. An *action*, a, in ADL is a set of state pairs $\prec s, t \succ$ where action a can be executed in state s to produce state t. The association between a state s and state description ϕ is indicated by the "models" symbol $\models$ and is written $s \models \phi$. An *action schema* in ADL, which more closely resembles the add and delete lists of STRIPS, characterizes a set of possible actions. Schemata are described by four optional groups of clauses:

1. Precond, the preconditions,
2. Add and Delete, a set of formulae describing the set of tuples to be added (deleted) to the interpretation of some relations R in the resulting state t, and
3. Update, a set of relations describing how functions change in t from s.

Pednault's version of our `MovB(m,l)` schema is then:

$$
\begin{array}{ll}
\texttt{MovB(l)} & \\
\quad\textsc{Add}: & \texttt{At(B,l)} \\
 & \texttt{At(z,l)}\ \forall \texttt{z} \mid \texttt{In(z)} \wedge \texttt{z} \neq \texttt{B} \\
\quad\textsc{Delete}: & \texttt{At(B,m)} \\
 & \texttt{At(z,m)}\ \forall \texttt{z} \mid \texttt{In(z)} \wedge \texttt{z} \neq \texttt{B}
\end{array}
$$

We merge the two Add and Delete categories into a single Effects category. The notion of "adding" a tuple for $R()$ is encoded as before. Deleting a tuple from $R()$ is encoded by asserting $\neg R()$. We note that this technique is merely a syntactic variant of Pednault's action representation. In this paper we also forbid the use of the UPDATE category as well as disjunction in preconditions.[5] We insist that all necessary preconditions be explicitly stated. This diverges from Pednault's [1986] approach, where he assumed preconditions could be inferred from the world state. Free variables in schemata are implicitly existentially quantified, serving as placeholders for future interpretations. Finally, equality constraints can only involve variables or variables and constants. All constants are assumed unique; their value cannot be changed through actions.

2.2 Secondary preconditions

Pednault represents action schemata internally as sets of *causation* and *preservation* preconditions collectively known as "secondary preconditions." Intuitively, a causation precondition of an action A for some relation R, Σ_R^A, specifies all conditions under which A will cause R to be added to the world state. The preservation preconditions of an action A and relation R, Π_R^A completely specify the conditions under which R is *not* deleted from the world state; hence, the conditions that "preserve" the truth value of R. For example, the following is a preservation condition for $At(u,v)$ with respect to action `MovB(m,l)`:

$$\Pi_{At(u,v)}^{MovB(m,l)} \equiv \neg[(u = B \wedge v = m) \vee (u \neq B \wedge v = m \wedge In(u))]$$

The first disjunct above corresponds to moving the briefcase from location `m`, thus affecting $At(\mathtt{B},\mathtt{m})$. The second disjunct corresponds to moving any item `z` that is in the briefcase, thus affecting $At(\mathtt{z},\mathtt{m})$. Ensuring that neither disjunct holds true thus preserves the value of $At(u,v)$ whenever a `MovB` is executed.

McDermott [1991] represented action schema as secondary preconditions to create a provably complete, total-order planner for ADL. While generating these secondary preconditions is straightforward (given the techniques found in Pednault's thesis [1986]) the resulting expressions are often unwieldy, containing multiple disjunctions, equality constraints and functionals. McDermott simplified these formulae through the use of heuristic rules for efficiency purposes. The resulting algorithm, however, sacrificed completeness and produced anomalous behavior for some simple tasks.

We chose a different approach. Our key insight was to first separate the logical connectives and relations from the metric functions and equality constraints. Next, we realized that generating a complete, disjunctive, secondary precondition was unnecessary in many circumstances. For example, not all disjuncts might be required for a particular planning task. We then devised an algorithm that constructed the secondary preconditions dynamically, introducing constraints and logical connectives only when absolutely necessary.

[5] While our implementation now successfully handles disjunctive preconditions and we have almost finished implementing a class of metric updates, we have not extended our proofs to these cases.

2.3 Steps and Effects

Instead of representing an action schemata as a complex set of secondary preconditions, we convert action schemata into canonical tuples called *steps* and *effects.* These tuples permit UCPOP to quickly identify the relevant parts of each add or delete condition when generating (or extending) a secondary precondition. We believe this results in a considerably more efficient regression process (see the comparison to PEDESTAL in section 6).

Definition *A* STEP *S is a triple $\prec\rho,\varepsilon,\beta\succ$ where ρ, the step preconditions, is a set of quantified literals; ε, the step effects, is a set of effects; and, β, the step constraints, is a set of equality constraints on free variables in ρ.*

The notations ρ_S, β_S and ε_S refer to the sets ρ, β and ε within step S. ρ_S are the common preconditions among all effects ε. The equality constraints in β apply to all formulae used in the effects ε as well as the preconditions ρ. The binding constraints β are represented as a set of (possibly negated) pairs, either (u,v) or $\neg(u,v)$. The former indicates that the free variables (or constants) u and v must codesignate, *i.e.*, u and v unify in any well-formed formula. The latter, $\neg(u,v)$ indicate non-codesignation, *i.e.*, u and v can not unify in any well-formed formula. The set of bindings β thus specify an equivalence relation (written "$\approx_\beta$") on plan variables and constants. We will say that a pair (u,v) is CONSISTENT WITH A SET OF BINDINGS β when $u \approx v$ holds true under the relation $\approx_\beta$. Similarly, $\neg(u,v)$ is consistent when $u \approx v$ does not hold true under $\approx_\beta$.

Each effect $e \in \varepsilon_S$ is then represented as follows. The notations ρ_E, β_E and θ_E refer to the sets ρ, β and θ within effect E.

Definition *An* EFFECT *E is a triple $\prec\rho,\theta,\beta\succ$ where ρ, the effect preconditions, is a set of quantified literals; θ, the effect postconditions, is a set of quantified literals; and β, the effect binding constraints, is a set of equality constraints on free variables in ρ and θ.*

The step and effect tuples clearly distinguish between the logical connectives and equality constraints required by secondary preconditions. When the UCPOP algorithm wants to generate part of a causation precondition for some relation R, it looks for an effect tuple $e = \prec\rho,\theta,\beta\succ$ where $\theta \vdash R$. Recall that the causation precondition Σ_R^A dictates *all* conditions under which an action A will cause R to become true. Since the effect e captures at least one possible way in which the action A can generate R, (*i.e.*, $\theta_e \vdash R$), it must be true that $\rho_e \wedge \beta_e \vdash \Sigma_R^A$. UCPOP can then post the conjuncts of ρ_e as new subgoals and β_e as new constraints that must be met by the plan. If the action A were newly instantiated from an action schema into a step s, UCPOP would also post the subgoals ρ_s and β_s. These new goals represent a nondeterministic choice (*i.e.*, the choice of one disjunct of Σ_R^A), corresponding to the decision to make sure that effect e will be the true cause of R. If this later proves impossible, UCPOP backtracks.

For example, consider a step $\prec\rho, \varepsilon, \beta\succ$ describing the action `MovB(m,l)`. The precondition, $\rho = \{\texttt{At(B,m)}\}$, specifies that the briefcase must first be at some location `m`. The binding constraints, $\beta = \{\neg(\texttt{m},\texttt{l})\}$, require that the origin and destination be distinct. The effects, ε, specify that both the briefcase and its contents move after execution of `MovB(m,l)`. This set ε is written below, following the internal representation of effects as tuples of the form $\prec\rho, \theta, \beta\succ$:

$$\left\{\begin{array}{l} \forall \texttt{z} \prec\{\texttt{In(z)}\}, \{\neg\texttt{At(z,m)}, \texttt{At(z,l)}\}, \{\neg(\texttt{z},\texttt{B})\}\succ, \\ \prec\emptyset, \{\neg\texttt{At(B,m)}, \texttt{At(B,l)}\}, \emptyset\succ \end{array}\right\}$$

Note the use of the quantifier $\forall$`z` to indicate that the free variable `z` in the first effect tuple is to be universally, not existentially, quantified.

3 Representing Plans and Problems

With a totally ordered sequence of steps, it is fairly easy to identify the first step that causes some proposition c to be true in the world. We call this step the SOURCE OF c; if c is true in the initial world state, we refer to the dummy step, S_0, as the source. With partially ordered steps, things are more complex and it is useful to explicitly record the intended source for a proposition during planning. Elaborating on the work of Tate [1977], McAllester [1991], and others we define:

Definition *A* CAUSAL LINK *is a quadruple* $\prec S_i, e, r, S_j \succ$, *denoted* $S_i \overset{e,r}{\rightarrow} S_j$, *where r is a (possibly negated) precondition of S_j (or a precondition of one of its effects), and e is an effect of S_i, and $\exists q \in \theta_e$ such that q unifies with r.*

To aid in its decision-making, UCPOP maintains a list of causal links from effects of steps to goals and subgoals. These links represent the assumptions upon which a plan P relies and are a crucial aspect of partially ordered PLANs.

Definition *A* PLAN *is quadruple* $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$, *where $\mathcal{S}$ is a set of steps, $\mathcal{B}$ is a set of binding constraints on free variables in $\mathcal{S}$, $\mathcal{O}$ is a set of ordering constraints $\{S_i < S_j \mid S_i, S_j \in \mathcal{S}\}$, and $\mathcal{L}$ is a set of causal links.*

Pednault's plans [1986] only included steps and ordering constraints. Our representation denotes the importance of separating the equality constraints from all other logical connectives. The equality constraints from secondary preconditions are gathered and kept in a single set, $\mathcal{B}$. No goals are ever posted for satisfying equality constraints. Instead, they are immediately checked against $\mathcal{B}$ for consistency. If at any time during the planning process this set becomes inconsistent, the plan is eliminated from consideration; UCPOP then backtracks. Plan tuples can be used to represent the problem being solved as we now explain.

Definition *A* PLANNING PROBLEM α *is a quadruple* $\prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *where Λ is a set of action schemata, $\mathcal{I}$ is a set of literals indicating the "initial conditions," Γ is a set of quantified clauses indicating the "goals," and U is a universe of discourse for variables in $\Lambda, \mathcal{I}$, and Γ.*

To assure systematic establishment of goals and subgoals that have universal quantified clauses, we map these clauses into a set of corresponding ground clauses.

Definition *The* UNIVERSAL BASE Υ *of a first-order clause is defined recursively as follows:*

- $\Upsilon(\Delta) = \Delta$ *if Δ contains no quantifiers.*
- $\Upsilon(\exists x_1 \ldots x_n \Delta) = \exists x_1 \ldots x_n \Upsilon(\Delta)$,
- $\Upsilon(\forall x_1 \ldots x_n \exists y_1 \ldots y_n \Delta) = \exists y'_1 \ldots y'_n \Upsilon(\Delta')$ *where Δ' is a conjunction of formulae identical to Δ, one for each possible interpretation of the x_i under a universe U. The y_i are renamed to unique names y'_i.*

For example, suppose that $\Delta = \{\exists x Above(x, A), \forall y \exists w On(y, w)\}$. Let the universe of discourse for Δ be the pair of blocks, $\{A, B\}$. The universal base $\Upsilon(\Delta)$ is the set of three elements:

$$\{\exists x Above(x, A),\ \exists w' On(A, w'),\ \exists w' On(B, w')\}$$

Each clause within $\Upsilon(\Delta)$ is said to be "universally ground." This reformation of universally quantified clauses is similar to the use of Skolem functions, $y = f(x)$, for existentials y and universals x. Recall that Skolem functions eliminate universals x by replacing existentials y with dummy functions $f(x)$ that produce the corresponding values for each instance of a universal. Whereas Skolem functions are applied from the inside out, the universal base is applied from the outside in.

The universal base of a set of clauses is isomorphic to the image of all Skolem functions over some set of clauses Δ. Instead of using $y = f(x)$, we enumerate the set $\{f(x_0), f(x_1), \ldots, f(x_n)\}$ for each x_i in the range of $f(x)$ and then generate the appropriate set of clauses Δ' through substitution on the corresponding universal x, then rename y to y'. Renaming prevents name conflicts in alternating quantifiers. We assume that the universe, U, and hence the universal base is finite.

Definition *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ denote a planning problem. The* GOAL PLAN *of α, written* g-plan(α)*, is a plan $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ where $\mathcal{S} = \{S_0, S_\infty\}$, $\mathcal{O} = \{S_0 < S_\infty\}$, $\mathcal{B} = \mathcal{L} = \emptyset$, the initial step, S_0 introduces α's initial conditions but has no preconditions or constraints and the goal step, S_∞, has α's goals as preconditions, but has no effects or binding constraints.*

In the next section, we describe an algorithm, UCPOP, that takes the goal plan of a problem as input and systematically adds steps and constraints until it finds one that solves the planning problem. In section 5 we prove that UCPOP is sound and complete.

4 The UCPOP Planning Algorithm

At the very heart, UCPOP is a theorem prover that resolves step preconditions against effect postconditions. It reduces all quantified formulae into universally ground propositions and then manipulates them to create and extend secondary preconditions. These preconditions are then "satisfied" by creating causal links and then protecting them from effects of other steps. A common subroutine of UCPOP is used throughout the algorithm:

Definition MGU(p, q) *is a function that returns the most general unifier of literals p and q. $\perp$ is returned if no such unifier exists.*

When Δ is a set of clauses, the notation $\Delta \backslash \mathtt{MGU}(p, q)$ represents the set of clauses Δ' resulting from applying the substitution $\delta \backslash \mathtt{MGU}(p, q)$ to every clause $\delta \in \Delta$. The form of a general unifier is taken to be a set of pairs $\{(u, v)\}$, indicating that u and v must codesignate to ensure that p and q unify. This will allow us to treat binding constraints β as a conjunction of general unifiers.

For example,

$$\begin{aligned} \mathtt{MGU}(In(P), In(x)) &= \{(x, P)\} \\ \mathtt{MGU}(In(D), In(D)) &= \emptyset \\ \mathtt{MGU}(At(x, y), In(x)) &= \perp \end{aligned}$$

4.1 Algorithm overview

The UCPOP algorithm (figure 2) takes three inputs: a plan $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$, an agenda of outstanding goals G, and a set of action schemata Λ. Each entry on the goal agenda is a pair, $\prec c, S\succ$, in which S denotes a plan step and c denotes a precondition of that step [Pednault 1986]. If c is an equality constraint on variables (u, v), we rename the variables to (u_s, v_s) and then add the

Algorithm UCPOP$(P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ, G, \Lambda)$

1. **Termination:** If G is empty, report success and return P.
2. **Goal selection:** Choose a goal $\prec c, S\succ \in G$. If a link $S_i \overset{e,\neg c}{\rightarrow} S$ exists in $\mathcal{L}$, fail (an impossible plan). Note that c is universally ground.
3. **Operator selection:** Nondeterministically choose any existing (from $\mathcal{S}$) or new (instantiated from Λ) step S_s with effect e and a universally ground clause $p \in \Upsilon(\theta_e)$ where $\mathtt{MGU}(c, p) \neq \perp$. If no such choice exists then fail. Otherwise, let $\mathcal{L}' = \mathcal{L} \cup \{S_s \overset{e,c}{\rightarrow} S\}$, $\mathcal{B}' = \mathcal{B} \cup \mathtt{MGU}(c, p) \cup \beta_e \cup \beta_{S_s}$, $\mathcal{O}' = \mathcal{O} \cup \{S_s < S\}$, $G' = G - \prec c, S\succ$, and let $\mathcal{S}' = \mathcal{S} \cup \{S_s\}$.
4. **Subgoal generation:** If effect e has not already been used to establish a link in $\mathcal{L}$ with bindings $\mathtt{MGU}(c, p)$ then let $G' = G$ and for each $\sigma \in \Upsilon(\rho_e \backslash \mathtt{MGU}(c, p))$ add $\prec\sigma, S_s\succ$ to G'. If $S_s \notin \mathcal{S}$, for each $\sigma \in \Upsilon(\rho_S \backslash \mathtt{MGU}(c, p))$ also add $\prec\sigma, S_s\succ$ to G'.
5. **Causal link protection:** Let l be a causal link $S_i \overset{e_i,q}{\rightarrow} S_j$ in $\mathcal{L}$. Let S_k be any step with an effect e_k and postcondition $p \in \theta_{e_k}$. Step S_k THREATENS link l with clause $v_p \in \Upsilon(p)$ if possibly $S_i < S_k < S_j$ when $\mathtt{MGU}(\neg q, v_p) \neq \perp$ is consistent with $\mathcal{B}$. For all such S_k, e_k, l and v_p such that S_k threatens l with v_p, nondeterministically do one of the following (or, if no choice exists, fail):
   (a) **Promotion** If possibly $S_j < S_k$, let $\mathcal{O}' = \mathcal{O} \cup \{S_j < S_k\}$.
   (b) **Demotion** If possibly $S_k < S_i$, let $\mathcal{O}' = \mathcal{O} \cup \{S_k < S_i\}$.
   (c) **Separation** Let $\mathcal{O}' = \mathcal{O} \cup \{S_i < S_k < S_j\}$ then nondeterministically
      i. Choose constraints β' on existentially quantified variables such that $\mathtt{MGU}(\neg q, v_p) = \perp$ and let $\mathcal{B}' = \mathcal{B}' \cup \beta'$, or
      ii. Choose a precondition $r \in \Upsilon(\rho_{E_k} \backslash \mathtt{MGU}(\neg q, v_p))$ and let $G' = G' \cup \{\prec\neg r, S_k\succ\}$.
6. **Recursive invocation:** If $\mathcal{B}'$ is inconsistent then fail; else call UCPOP$(\prec\mathcal{S}', \mathcal{B}', \mathcal{O}', \mathcal{L}'\succ, G', \Lambda)$.

Figure 2: The UCPOP partial order planning algorithm handles actions with universally quantified preconditions and effects as well as conditional effects yet is sound and complete.

constraint to $\mathcal{B}$ (constants remain unchanged). Variable renaming is done when an action schema is first instantiated as a step in the plan (line 3 of UCPOP), following the use of "fresh variables" in [McAllester and Rosenblitt 1991].

When used at the top level to solve a planning problem $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$, the algorithm is called with α's goal plan, its universally ground goals, and its schemata: UCPOP(`g-plan`(α), $\Upsilon(\Gamma), \Lambda$). Note that while the goal agenda G is a set of tuples $\prec c, S\succ$ where c must be achieved by step S, we sometimes use an abbreviation for top-level goals. For example, we often write $\Upsilon(\Gamma)$ rather than the more cumbersome

$$\{\prec c, S_\infty\succ \mid \forall c \in \Upsilon(\Gamma)\}$$

Our algorithm depends on the following assumptions:

- The initial world state is complete.
- The universe of discourse U is fixed and finite.
- All changes to the world state are dictated by actions and explicitly stated.
- Actions are deterministic.
- Actions are consistent, *i.e.*, no action will add both ϕ and $\neg\phi$ to any consistent world state under any condition.
- All relations R are *regressively ascertainable everywhere* [Pednault 1986], *i.e.*, the truth value of every relation R at some step S_i can be determined solely by the actions and the initial state.

The UCPOP algorithm continually refines an incomplete plan P until all goals and subsequent subgoals are satisfied. These refinements include the addition of new steps to $\mathcal{S}$, the constraining of free variables through additional codesignation bindings in $\mathcal{B}$, and the ordering of steps via constraints in $\mathcal{O}$. Lines 3 and 4 dynamically introduce portions of causation preconditions. Line 5 chooses how to create disjuncts of preservation conditions for some relation q. These rely heavily on the structure of effects and steps.

4.2 Efficiency concerns

The UCPOP algorithm is clearly exponential. To improve efficiency, the UCPOP implementation distinguishes between static and dynamic propositions and avoids copying the static world description in each partial plan. The universe of discourse is implemented as a static, typed hierarchy of LISP objects. Universal quantification is handled "lazily," using an iteration abstraction that dynamically expands universal clauses to cover more and more cases (similar to the plan transformation rule 4.11 in [Pednault 1986]). A closed world assumption is adopted to prevent a large ε_{S_0} that would normally have several $\neg p$ effects. The codesignation constraints β are implemented as monotonically converging equivalence classes, similar to the union-find algorithm. Partial orderings on steps are simple lists of pairs $(S_i, S_j) \iff S_i < S_j$, a departure from the current trend of complex temporal databases. UCPOP uses A^* search (with the same evaluation function) on every space of partial plans. Although there are numerous hooks for domain dependent heuristics, we haven't yet systematically explored their use; we believe that this avenue offers the greatest opportunity for performance improvement.

5 Formal Properties

In this section we prove that UCPOP is sound and complete. We adopt the model-theoretic semantics of ADL and refer the reader to [Pednault 1986] for a complete description.

5.1 Formalizing Solutions

Soundness implies that every plan produced by UCPOP is a solution to the original problem. Completeness implies that, if there is a solution to some problem α, UCPOP will find it. Both of these concepts rely on the definition of a SOLUTION. We construct this definition by starting from notions of world states, then defining what it means to execute actions, execute plans, and finally converging on the formal concept of solution.

Recall that states s of the world are algebraic structures, *i.e.*, models in logic. Steps, which are instances of ADL action schemata, are modeled by a set of state pairs $\prec s, t\succ$ where the step can be executed in state s to produce state t. A set of types T describes collections of objects in U. Each type $t_i \in T$ is a fixed set of objects $\{o_0, o_1, \ldots, o_n\} \subseteq U$. Types may overlap; for example, `block` might be a type representing all known blocks in U and `physob` might be a set of all physical objects. Types play an important role when determining the scope of universal quantifiers.

We use a primitive state-based model of time where all actions are atomic and cannot overlap. Thus, the temporal history of the world is represented by a linear sequence of states separated by single actions.

Given these concepts, we now define what it means to execute a step. We have slightly changed the definition in [Pednault 1986] to correspond to our notation and assumptions. Deterministic actions insist that every state s exists in only one pair $\prec s, t\succ$ in an action a. Whereas Pednault assumed a set of initial states, we assume one complete initial state.

Definition *Let $\{S_i\}_{i=0}^n$ be a sequence of steps S_i and let $\mathcal{W}$ be a state of the world. The* RESULT OF EXECUTING $\{S_i\}_{i=0}^n$ IN $\mathcal{W}$, *written* RESULT$(\{S_i\}_{i=0}^n \mathcal{W})$, *is defined recursively as follows:*

- RESULT$(\{S_i\}_{i=0}^0, \mathcal{W}) = \mathcal{W}$
- RESULT$(\{S_i\}_{i=0}^n, \mathcal{W}) = t$ *such that*

$\prec$Result$(\{S_i\}_{i=0}^{n-1}, \mathcal{W}), t\succ$ *is a state pair in the action* S_n.

The previous definition says nothing about whether it is feasible to "execute" an action in some state. It only details what happens to a state s when the action is applied to produce state t. We now define the notion of executability for partially ordered plans. First, we define a mapping from partially ordered to totally ordered plans:

Definition *Let* $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *denote a planning problem. A totally ordered sequence of steps* $\{S_i\}_{i=0}^{n}$ *is a* GROUND TOPOLOGICAL SORT *of a plan* $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ *if there is a bijection,* f*, from* $\mathcal{S}$ *to the steps in* $\{S_i\}_{i=0}^{n}$*,* f *is a homomorphism with respect to the ordering constraints* $\mathcal{O}$*, there exists a global substitution* Θ *that binds all unbound variables and is consistent with* $\mathcal{B}$*, and for every step* S *in the plan,* $f(S) = S\backslash\Theta$.

Given this, we can map partially ordered plans into sets of totally ordered plans, and then define what it means to execute each one:

Definition *A step* $A = \prec\rho, \varepsilon, \beta\succ$*, representing a set of possible actions, is* EXECUTABLE IN A STATE s *if and only if* $s \models \rho_A$ *and* β_A *is consistent with* s*. A plan* $P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ *is* EXECUTABLE IN INITIAL STATE $\mathcal{I}$ *if, for every ground topological sort* $\{S_i\}_{i=0}^{n}$ *of* P*, each step* S_i *is executable in state* Result$(\{S_i\}_{i=0}^{i-1}, \mathcal{I})$.

Recall that a planning problem is a collection of action schemata, initial conditions, universe, and goals.

Definition *A plan* $Q = \prec\mathcal{S}_q, \mathcal{B}_q, \mathcal{O}_q, \mathcal{L}_q\succ$ *is an* ENHANCEMENT *of a plan* $P = \prec\mathcal{S}_p, \mathcal{B}_p, \mathcal{O}_p, \mathcal{L}_p\succ$ *for a planning problem* $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *if and only if* $\mathcal{S}_p \subseteq \mathcal{S}_q$*,* $\mathcal{O}_p \subseteq \mathcal{O}_q$*,* $\mathcal{B}_p \subseteq \mathcal{B}_q$*, and* $\mathcal{L}_p \subseteq \mathcal{L}_q$.

Definition *A* SOLUTION *to a planning problem* $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *is a plan* $P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ *where* P *is an enhancement of* `g-plan`(α)*,* P *is executable in* $\mathcal{W}$*, and all* $S_i \in \mathcal{S}$ *are instances of action schemata in* Λ.

Note that if P is a solution, it follows that Result$(\{S_i\}_{i=0}^{n}, \mathcal{I}) \models \Gamma$ since $\rho_\infty = \Gamma$ and S_∞ is executable.

5.2 Soundness

Our proof of soundness relies heavily on Pednault's causality theorem [1986] which is akin to a version of Chapman's modal truth criterion [Chapman 1987] for plans with conditional actions. We restate the causality theorem here for convenience:

Theorem 1 Pednault's Causality Theorem. *A condition* φ *will be true at a point* p *during the execution of a plan if and only if one of the following holds:*

1. *An action* a *is executed prior to point* p *such that*
   (a) Σ_φ^a *is true immediately before executing* a.
   (b) Π_φ^b *is true immediately before the execution of each action* b *between* a *and point* p.
2. φ *is true in the initial state and* Π_φ^a *is true immediately before the execution of each action* a *prior to point* p.

Proving soundness means demonstrating that the UCPOP algorithm is correct, *i.e.*, that every answer from UCPOP(`g-plan`$(\alpha), \Upsilon(\Gamma), \Lambda$) is a correct solution to the planning problem.

We use a standard technique for recursive algorithms, proving that a loop invariant holds before and after every recursion. We use this to show that this invariant holds whenever UCPOP halts. The invariant we use is defined as follows.

Definition (The UCPOP Loop Invariant) *If the subgoals in the goal agenda* G *are satisfied by* P*, then* P *will be a solution to* α.

Lemma 2 *The* UCPOP *loop invariant holds on the initial call to* UCPOP(`g-plan`$(\alpha), \Upsilon(\Gamma), \Lambda$).

Proof. Trivially, if the goals $G = \Upsilon(\Gamma)$ are satisfied for step S_∞, then Γ is satisfied and P would be a solution. □

Lemma 3 *If the loop invariant holds before an iteration of* UCPOP*, it will hold afterwards.*

Proof. By corollary 3.29 of Pednault's thesis [1986], we can replace goals in G (a subset of his "agenda") with the causation preconditions of steps in P whose effects achieve those goals and the preservation preconditions of steps that might threaten the goals. Then, by Pednault's causality theorem [Pednault 1991], if these preconditions are satisfied, the original goals G are satisfied. We now argue that UCPOP correctly performs these goal transformations.

The condition φ and the "point p" of the causality theorem match the variable c and the step S from line 1 of UCPOP. The action a refers to the new or existing step S_s from line 2 of UCPOP. Note that the new steps S_s are instantiated from Λ and that this is the only place where new steps are introduced. Also, note that case 2 of the causality theorem is handled by choosing step S_0 in line 2 of UCPOP.

UCPOP diverges from a direct, procedural encoding of the causality theorem as a result of the following observation. Instead of requiring that the entire causation preconditions Σ_φ^a be generated and posted as a subgoal, it is sufficient to post another condition ρ that subsumes the more complex formulae, *i.e.*, where $\rho \vdash \Sigma_\varphi^a$. These ρ constraints are exactly those formulae stored in step and effect tuples which, mentioned earlier, are syntactic transformations of action schemata. Following the assumptions of UCPOP we can translate the causation preconditions from Pednault's thesis and rewrite them as follows:

$$\begin{aligned}
\Sigma^a_\varphi &= \bigvee_{\forall e \in \varepsilon_a \mid \theta_e \vdash \varphi} \rho_e \wedge \beta_e \\
\Sigma^a_{x=y} &= \text{False} \leftrightarrow \text{x and y are unique constants,} \\
&\quad\ \text{True otherwise.} \\
\Sigma^a_{x \neq y} &= \text{False} \leftrightarrow \text{x and y are the same constant,} \\
&\quad\ \text{True otherwise.}
\end{aligned}$$

The operator selection line of UCPOP chooses an effect e where $\texttt{MGU}(c,p) \neq \perp$ and $p \in \Upsilon(\theta_e)$. Since $\texttt{MGU}(c,p)$ returns a general unifier and $\Upsilon(\theta_e)$ returns a set of formulae p such that $\theta_e \vdash p$, if $\texttt{MGU}(c,p) \neq \perp$ then $\theta_e \vdash c$. Thus, substituting φ for c, UCPOP correctly chooses one disjunct, for effect e, of the secondary precondition Σ^a_φ. Since UCPOP works with action schema, it dynamically adjusts the set of actions a represented by the schema by also introducing the constraints $\texttt{MGU}(c,p)$. Following corollary 3.29, UCPOP replaces the original goal c with ρ_e (line 4) and records this decision as a causal link $S_s \xrightarrow{e,c} S$.

Equality constraints $a = b$ for the step S, which are derived by a disjunction of β_e's, are handled as follows. UCPOP renames the variables to $a_s = b_s$ and then adds them to $\mathcal{B}$ (line 3, $\mathcal{B}' = \mathcal{B} \cup \texttt{MGU}(c,p) \cup \beta_e \cup \beta_{S_s}$). If $\mathcal{B}$ ever becomes inconsistent, at least one of the equality "goals" is not met. This mimics the secondary precondition $\Sigma^a_{x=y} = \text{False}$. Otherwise, any grounding of the free variables in a consistent set of constraints $\mathcal{B}$ would satisfy the equality goals, subsuming $\Sigma^a_{x=y} = \text{True}$. Thus $\mathcal{B}$ is a syntactic translation of goals of the form $a = b$ at some step S and UCPOP correctly posts this subset of the causation preconditions for step S_s and effect e. This depends upon our assumption that all free variables are existential and that all constants are unique. We conclude that UCPOP correctly handles case (1), subcase (a) of the causality theorem.

Case (1), subcase (b) and case (2) of the causality theorem require preservation preconditions for all steps b that possibly come between step S_s and S in the plan that requires c to be true at S. Again, we rewrite these preconditions in terms of our assumptions and tuples, as follows:

$$\begin{aligned}
\Pi^a_\varphi &= \bigwedge_{\forall e \in \varepsilon_a \mid \theta_e \vdash \varphi} \neg\rho_e \vee \neg\beta_e \\
\Pi^a_{x=y} &= \text{False} \leftrightarrow \text{x and y are the same constant,} \\
&\quad\ \text{True otherwise.} \\
\Pi^a_{x \neq y} &= \text{False} \leftrightarrow \text{x and y are unique constants,} \\
&\quad\ \text{True otherwise.}
\end{aligned}$$

UCPOP handles subcase b as follows. It either (1) introduces additional step ordering such that subcase b no longer applies, or (2) follows the previous intuition and posts goals $\neg q$ such that $\neg q \vdash \Pi^a_\varphi$ for all such actions a. Lines 5a and 5b of UCPOP handle the former approach. Line 5c, in conjunction with the conditions of line 5, handle the latter (following the previous argument for causation preconditions).

By [Pednault 1986, corollary 3.29], satisfying the newly revised goal agenda G' is equivalent to satisfying G. UCPOP only recurses when $\mathcal{B}'$ is consistent and it only introduces new steps from Λ. Since the loop invariant holds before the execution of UCPOP every ground topological sort of the original plan P would be executable if G were satisfied. By corollary 3.29 and the fact that the action preconditions for S_s are introduced in line 4, P' must also be executable if the new G' were satisfied. Since UCPOP monotonically increases the set of binding constraints, causal links and steps, each iteration returns an enhancement of $\texttt{g-plan}(\alpha)$. Thus, P would be a solution if G' were true. □

Lemma 4 *If the loop invariant holds before an iteration of* UCPOP *and then* UCPOP *halts, the invariant still holds.*

Proof. UCPOP halts in line 1 when the goal agenda G is empty, without performing any modifications. Since no structures are modified the loop invariant must still hold after UCPOP halts.

Theorem 5 (Soundness) *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ be a planning problem. If* $\text{UCPOP}(\texttt{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$ *returns a plan P then P is a solution for α.*

Proof. The previous lemmas combine to form a simple inductive argument that the UCPOP loop invariant holds whenever UCPOP halts provided that it is invoked with $\text{UCPOP}(\texttt{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$. Since the goal agenda is empty when UCPOP halts, no additional conditions are required for P to be a solution. Thus, any plan P returned by $\text{UCPOP}(\texttt{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$ must be a solution to α in and of itself. □

5.3 Completeness

Before stating the completeness theorem, we present a few useful corollaries of the soundness theorem.

Corollary 5.1 *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ be a planning problem. If $\{S_i\}_{i=0}^n$ is a solution to α then for $2 < k < n$, $\{S_i\}_{i=k}^n$ is a solution to $\prec\Lambda$, $\text{Result}(\{S_i\}_{i=0}^{k-1}, \text{I})$, U, $\Gamma\succ$.*

In other words: if an n step plan is a solution to a planning problem with initial conditions $\mathcal{I}$, then the last $n-1$ steps are a solution to the modified planning problem which starts with the initial conditions derived by executing the first step in $\mathcal{I}$.

Proof. Follows directly from the definition of $\textsc{Result}(\{S_i\}_{i=0}^n, \mathcal{I})$. □

Definition *A plan $P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ is* FULLY SUPPORTED *if every step $S_i \in \mathcal{S}$ is fully supported. A step S is* FULLY SUPPORTED *in P if its preconditions ρ_S are fully supported. A precondition r of a step S_j is* FULLY SUPPORTED *in P if there is a causal link $S_i \stackrel{e,r}{\rightarrow} S_j \in \mathcal{L}$, $S_i < S_j \in \mathcal{O}$ and effect e is fully supported. An effect e of step S is* FULLY SUPPORTED *in P if S is fully supported and every precondition $r \in \rho_e$ is fully supported.*

Corollary 5.2 *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ be a planning problem. If* UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ) *returns a plan P then P is fully supported.*

Proof. Follows from the soundness theorem. Note that each link (line 3) records decisions made by UCPOP as to which steps and effects satisfy which preconditions. □

Theorem 6 (Completeness) *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ be a planning problem and let $\{S_i\}_{i=0}^n$ be a solution to* α with no extra steps. *With the appropriate search strategy* UCPOP *applied to* g-plan(α) *will return a solution, P, such that $\{S_i\}_{i=0}^n$ is a ground topological sort of P.*

Proof. We use induction on the number of steps in the totally ordered solution. To finesse issues of search control, we use $\{S_i\}_{i=0}^n$ as an oracle to guide the construction of the partially ordered plan; McDermott [1991] refers to this as a clairvoyant algorithm. A* or IDA* search [Korf 1985] is used to maintain completeness in our implementation.

Base Case: $\{S_i\}_{i=0}^k \wedge k = 2$.

This means that no plan steps (besides the dummy initial and goal steps) are necessary to achieve the goal, *i.e.*, $\mathcal{I} \vdash \Gamma$. Thus a call to clairvoyant UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ) will result in operator selection (step 3) consistently choosing S_0 as the establishing step and making causal links solely to this initial step. Since no new steps are added, none of the links can be threatened. Thus UCPOP will terminate with a solution, P, that has no additional steps, satisfying our need to have the sequence be a ground topological sort of P.

Inductive Step: $\{S_i\}_{i=0}^k \wedge k > 2$.

Given a solution $\{S_i\}_{i=0}^n$ with at most $k-1$ steps, *i.e.*, $n < k$, we assume inductively that clairvoyant UCPOP correctly generates a solution, P, such that $\{S_i\}_{i=0}^n$ is a ground topological sort of an enhancement of P. Now consider the solution $\{S_i\}_{i=0}^k$ to the planning problem $\prec\Lambda, \mathcal{I}, U, \Gamma\succ$; we show that UCPOP finds a corresponding solution here as well.

Let α' be the planning problem $\prec\Lambda$, Result(S_1, $\mathcal{I}$), U, $\Gamma\succ$. By inductive assumption and corollary 5.1, clairvoyant UCPOP(g-plan(α'), $\Upsilon(\Gamma)$, Λ) will return a solution $P_a = \prec\mathcal{S}_a, \mathcal{B}_a, \mathcal{O}_a, \mathcal{L}_a\succ$ when given the $k-1$ step sequence $\{S_i\}_{i=2}^k$ as guidance. P_a is very close to the plan that we are seeking, but the initial step of P_a is doing double duty — it is acting as the source for propositions that either S_0 or S_1 provide in the original totally ordered solution. To distinguish these, we define three subsets of the causal links in $\mathcal{L}_a$.

We first denote a set L_0 of links whose source propositions c remain unchanged from the original initial conditions $\mathcal{I}$ until after the step S_1 is executed. Thus, S_1 does not delete any of these propositions (although it may add duplicates which are absorbed by $\textsc{Result}(S_1, \mathcal{I})$):

- $L_0 = \{S_i \stackrel{e,c}{\rightarrow} S_j \mid i = 0 \wedge \mathcal{I} \models c \wedge \textsc{Result}(S_1, \mathcal{I}) \models c\}$.

Another, distinct subset of links L_{add} contains links whose source propositions must be added by the execution of step S_1. If a proposition c is not in the initial conditions $\mathcal{I}$, but is in $\textsc{Result}(S_1, \mathcal{I})$, and since S_1 is the only step executed between these two situations, c can only be caused by some effect e in S_1:

- Let $L_{add} = \{S_i \stackrel{e,c}{\rightarrow} S_j \mid i = 0 \wedge \mathcal{I} \not\models c \wedge \textsc{Result}(S_1, \mathcal{I}) \models c\}$

A third set L_{del} contains links whose source propositions must be "deleted" by the execution of step S_1. If a proposition c was contained in the initial conditions $\mathcal{I}$, but is absent in $\textsc{Result}(S_1, \mathcal{I})$, then c must be deleted by one of S_1's effects since S_1 is the only step executed between these two situations:

- Let $L_{del} = \{S_i \stackrel{e,c}{\rightarrow} S_j \mid i = 0 \wedge \mathcal{I} \models c \wedge \textsc{Result}(S_1, \mathcal{I}) \not\models c\}$

Now consider the execution trace of all choices made by UCPOP in constructing P_a. We can use this trace, in conjunction with the above sets of links, to carefully guide UCPOP in solving the original problem α, producing a plan with $n+1 = k$ steps. The only difference between α' and α is in the initial state, $\mathcal{I}$. We will thus have to intervene, guiding UCPOP whenever it wants to create a link from the initial state. Otherwise, all goals and threat elimination can proceed as in the execution trace of P_a.

Whenever a link was chosen from L_0 for P_a, UCPOP is guided to create a link from the original, initial conditions captured by an effect in S_0. Whenever a link $S_0 \stackrel{e,c}{\rightarrow} S_j$ was chosen from L_{add} or L_{del} to create P_b, UCPOP is guided to create a corresponding link $S_1 \stackrel{e,c}{\rightarrow} S_j$ from a new step, S_1, which is identical to the step S_1 in the sequence $\{S_i\}_{i=0}^k$. All other links can proceed as before. Finally, after the execution trace from P_a is exhausted, UCPOP will have additional goals of the form

$\prec c, S_1 \succ$. Since S_1 was executable in $\text{RESULT}(S_0, \mathcal{I})$, these can be satisfied by links from the initial state $\mathcal{I}$ without posing new threats. UCPOP is then guided to create these links. The final result is an enhancement of g-plan(α) and it must executable in the initial state since it is nearly identical to P_a, except for the step S_1, which we know is executable. Thus, we have constructed a solution to α that contains a ground topological sort $\{S_i\}_{i=0}^{k}$. Thus, UCPOP is complete. $\square$

6 Related Work

Our work was directly motivated by that of Chapman [1987] and McAllester and Rosenblitt [1991] on the foundations of partial order STRIPS planning. The basic SNLP algorithm [Barrett *et al.* 1991], based on the work of McAllester, was derived from Chapman's TWEAK [1987] and Tate's NONLIN [1977]. These partial-order planners use action representations based on the STRIPS formalism. Chien [Chien and DeJong 1992] introduced conditionals into TWEAK and proved an incremental convergence to soundness. The work is parallel to our efforts on proving UCPOP sound.

Pednault provides an elegant theoretical foundation for total order planning with ADL in [Pednault 1986,Pednault 1988]. This work is extended to handle partial order plans in [Pednault 1991]. Although UCPOP was developed independently from [Pednault 1991], pages 243–244 of that document provide an informal description of lines 3–5 of our algorithm. Pednault's theory of planning and action transcend our implementation as it encompasses incomplete initial states, nondeterministic actions, functional updates, and disjunctive preconditions. While Pednault did pose a rule-based, plan enhancement algorithm, no complete implementation of his theory has yet been attempted. We have selected a large subset of this language, but are still far from total coverage.

McDermott's `Pedestal` [1991] was the first implementation of a planner using ADL, but `Pedestal` used a total order plan representation. McDermott proved a simple version of his algorithm complete, but frustration with performance issues led him to pursue heuristic variations that sacrificed completeness. We believe that the fundamental problem with `Pedestal` is its brute force generation of entire preservation and causation preconditions at runtime. Since these formulae tend to be fraught with redundant formulae and constraints, heuristic simplification is the only recourse.

Instead of generating the entire secondary preconditions as suggested by Pednault [Pednault 1986] and implemented by McDermott [1991], UCPOP divides these preconditions into separate logical, equality, (and in our current extensions) metric functional aspects; this allows specialized solvers to deal with the constraints in an optimized fashion.

Our work follows that of Collins and Pryor [1992] where they extended SNLP to handle conditional effects. They did not, however, consider universal quantification in either preconditions or effects nor did they prove soundness or completeness.

7 Future Plans

We have enhanced the algorithm to handle disjunctive preconditions and have almost completed the extension which allows UPDATES of monotonic metric functions. A high priority is to extend our formal results to the augmented algorithm. We also wish to allow actions that create and destroy objects in the universe of discourse U.

We are surprised by UCPOP's performance on the simple problems on which we have tested it and we plan to investigate performance issues more thoroughly. Doubtlessly, UCPOP's speed is largely a function of the simplicity of the classic test problems. Although we have described our algorithm nondeterministically, any actual implementation of it must search. It is manifest that good search techniques are the key to efficient planning. As a result, we believe that UCPOP affords an excellent platform for experimentation with search control heuristics [Barrett 1992,Knoblock 1991] and speed-up learning techniques [Minton 1988,Etzioni 1990b,Etzioni 1990a]. We have therefore developed a model and language for supporting search control heuristics in UCPOP (similar to that in PRODIGY [Minton *et al.* 1989]). Although fully implemented, our model for search control is still being tested; we plan to report our findings in the near future.

8 Conclusions

This paper presents a clean and elegant algorithm for partial order planning with an expressive action representation. UCPOP handles a large subset of ADL, including actions with conditional effects, universally quantified preconditions and effects, and universally quantified goals. We prove that UCPOP is sound and complete and briefly describe our full implementation. We believe that UCPOP's simplicity and efficient implementation makes it an excellent vehicle for further research on planning and learning.

ACKNOWLEDGMENTS

This research was funded in part by National Science Foundation Grant IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation. Ed Pednault made many detailed and helpful suggestions; we also acknowledge useful discussions with Tony Barrett, Benjamin Grosof, and Steve Hanks.

References

[Barrett and Weld 1992] A. Barrett and D. Weld. Partial Order Planning: Evaluating Possible Efficiency Gains. Technical Report 92-05-01, University of Washington, Department of Computer Science and Engineering, July 1992.

[Barrett *et al.* 1991] A. Barrett, S. Soderland, and D. Weld. The Effect of Step-Order Representations on Planning. Technical Report 91-05-06, University of Washington, Department of Computer Science and Engineering, June 1991.

[Barrett 1992] A. Barrett. Search-Control Heuristics and Abstraction in Least-Commitment Planning. In *Proceedings of the 1992 Workshop on Problem Reformulation and Representation Change.* NASA technical Report FIA-92-06, May 1992.

[Chapman 1987] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, July 1987.

[Chien and DeJong 1992] S. Chien and G. DeJong. Incremental Reasoning in Explanation-based Learning of Plans: A Method and Evaluation. In *Proceedings of AAAI-92*, August 1992.

[Collins and Pryor 1992] G. Collins and L. Pryor. Achieving the functionality of filter conditions in a partial order planner. In *Proceedings of AAAI-92*, August 1992.

[Etzioni 1990a] Oren Etzioni. *A Structural Theory of Explanation-Based Learning.* PhD thesis, Carnegie Mellon University, 1990. Available as technical report CMU-CS-90-185.

[Etzioni 1990b] Oren Etzioni. Why Prodigy/EBL works. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.

[Fikes and Nilsson 1971] R. Fikes and N. Nilsson. STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4), 1971.

[Kautz 1982] H. Kautz. A first order dynamic logic for planning. Tech Rept CSRG-144, Department of Computer Science, University of Toronto, 1982.

[Knoblock 1991] C. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, Carnegie Mellon University, 1991. Available as technical report CMU-CS-91-120.

[Korf 1985] R. Korf. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1), 1985.

[McAllester and Rosenblitt 1991] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of AAAI-91*, pages 634–639, July 1991.

[McDermott 1991] D. McDermott. Regression planning. *International Journal of Intelligent Systems*, 6:357–416, 1991.

[Minton *et al.* 1989] Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, 40:63–118, 1989. Available as technical report CMU-CS-89-103.

[Minton *et al.* 1991] S. Minton, J. Bresina, and M. Drummond. Commitment Strategies in Planning: A Comparative Analysis. In *Proceedings of IJCAI-91*, August 1991.

[Minton 1988] S. Minton. Quantitative Results Concerning the Utility of Explanation-Based Learning. In *Proceedings of AAAI-88*, pages 564–569, August 1988.

[Pednault 1986] E.P.D. Pednault. *Toward a mathematical theory of plan synthesis.* PhD thesis, Stanford University, December 1986.

[Pednault 1988] E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.

[Pednault 1989] E.P.D. Pednault. Adl: Exploring the middle ground between strips and the situation calculus. In *Proceedings Knowledge Representation Conf.,*, 1989.

[Pednault 1991] E.P.D. Pednault. Generalizing nonlinear planning to handle complex goals and actions with context-dependent effects. In *Proceedings IJCAI-91*, July 1991.

[Rosenschein 1981] S.J. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of IJCAI-81*, August 1981.

[Sussman 1975] G. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

[Tate 1977] A. Tate. Generating Project Networks. In *Proceedings of IJCAI-77*, pages 888–893, 1977.

[Weld and de Kleer 1989] D. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, San Mateo, CA, August 1989.