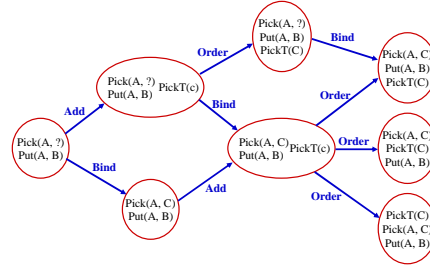


# Planning, Execution & Learning: Partial Order Planning

Reid Simmons

## Partial Order Planning

- Basic Idea
  - Search in plan space and use least commitment, when possible



## Plan-Space Search

- Search space is set of *partial plans*
- Plan is tuple  $\langle A, O, B \rangle$ 
  - $A$ : Set of **actions**, of the form  $\langle a_i : Op_i \rangle$
  - $O$ : Set of **orderings**, of the form  $\langle a_i < a_j \rangle$
  - $B$ : Set of **bindings**, of the form  $\langle v_i = C \rangle, \langle v_i \neq C \rangle, \langle v_i = v_j \rangle$  or  $\langle v_i \neq v_j \rangle$
- Initial plan:
  - $\langle \{start, finish\}, \{start < finish\}, \{ \} \rangle$
  - $start$  has no preconditions; Its effects are the initial state
  - $finish$  has no effects; Its preconditions are the goals

## Least Commitment

- Basic Idea
  - Make choices that are **only relevant to solving the current part of the problem**
- Least Commitment Choices
  - Orderings**: Leave actions unordered, unless they must be sequential
  - Bindings**: Leave variables unbound, unless needed to unify with conditions being achieved
  - Actions**: Usually not subject to "least commitment"
- Refinement
  - Only **add** information to the current plan
  - Transformational** planning can remove choices

## Plan Terminology

- Totally Ordered** Plan
  - There exists sufficient orderings  $O$  such that all actions in  $A$  are ordered with respect to each other
- Fully Instantiated** Plan
  - There exists sufficient constraints in  $B$  such that all variables are constrained to be equal to some constant
- Consistent** Plan
  - There are no contradictions in  $O$  or  $B$
- Complete** Plan
  - Every precondition  $p$  of every action  $a_i$  in  $A$  is **achieved**: There exists an effect of an action  $a_j$  that comes before  $a_i$  and unifies with  $p$ , and no action  $a_k$  that deletes  $p$  comes between  $a_j$  and  $a_i$

## Plan-Space, Partial-Order Planning

- General Approach
  - Find unachieved precondition
    - Add new action **or** link to existing action
  - Determine if conflicts occur
    - Previously achieved precondition is "clobbered"
    - Fix conflicts (reorder, bind, ...)
- Partial-order planning can easily (and optimally) solve the "Sussman Anomaly" problem



### POP and Sussman's Anomaly

1.

2.

3.

Planning, Execution & Learning: Partial Order 7 Simmons : Spring 2007

### POP and Sussman's Anomaly

4.

5.

Planning, Execution & Learning: Partial Order 8 Simmons : Spring 2007

### Modal Truth Criterion [Chapman, 1987]

- Modal Truth Criterion (MTC)
  - Formalized criterion for determining whether a (partial) plan achieves a given precondition  $p$  at a given step  $s$ 
    - $p$  is true in  $s$  if:
 
$$\exists t ((\Box < s) \wedge \Box \text{asserts}(t, p)) \wedge$$

$$\forall C ((\Box < C) \vee$$

$$\forall q ((\Box q \approx p) \Rightarrow \Box \neg \text{denies}(C, q) \vee$$

$$\exists W ((\Box C < W) \wedge (\Box W < s) \wedge$$

$$\exists r (\text{asserts}(W, r) \wedge (p \approx q) \Rightarrow (p \approx r))))))$$
- Can be used to generate planning algorithm (TWEAK)
  - step addition / establishment
  - promotion/demotion
  - separation
  - white knight

Planning, Execution & Learning: Partial Order 9 Simmons : Spring 2007

### SNLP [McAllester & Rosenblitt, 1991]

- Systematic Non-Linear Planner (SNLP)
  - Efficient way to determine which preconditions are achieved
  - Explore each node in search space at most once
    - Not clear whether this is an advantage...
- Causal Links
  - The "purpose" of an action (which condition it supports)
  - $a_i \rightarrow^c a_j$ , where  $a_i, a_j$  are actions and  $c$  is an effect of  $a_i$
  - Plan =  $\langle A, O, B, L \rangle$
- Threats
  - Action  $a_k$  with an effect  $c'$  that might "clobber" a causal link
  - Promotion**: Order  $a_k$  after  $a_i$
  - Demotion**: Order  $a_k$  before  $a_i$
  - Separation**: Constrain  $c'$  so that it does not unify with  $c$  (non-codesignation constraint)

Planning, Execution & Learning: Partial Order 10 Simmons : Spring 2007

### UCPOP [Penberthy & Weld, 1992]

- Universal, Conditional Partial-Order Planner (UCPOP)
  - Extension of SNLP to handle more expressive operators
    - Conditionals
    - Disjunction in preconditions
    - Universal and existential quantification
- Uses **unification** to find necessary bindings
  - Most General Unifier:  $\text{MGU}(p, q, B) = \{(v_i, x_i), \dots\}$
- Uses **constraint satisfaction** to prove consistency of plans
  - Consistent orderings
  - Consistent variable bindings (co-designation)

Planning, Execution & Learning: Partial Order 11 Simmons : Spring 2007

### UCPOP Language Extensions

- Conditionals
  - $(\text{when } (?b \neq \text{table}) (\text{clear } ?b))$
  - Add a new threat resolution mechanism: **confrontation**
    - Add the **negation** of conditional effect antecedent to the set of goals that must be achieved
- Disjunction in Preconditions
  - Add a new choice point to the algorithm that non-deterministically chooses to achieve one of the disjuncts
- Quantification
  - Typed formula:  $(\text{forall } \langle \text{type} \rangle \langle \text{var} \rangle \langle \text{expression} \rangle)$
  - Universal**: Expand into equivalent conjunct (assumes finite, known universe of objects)
  - Existential**: Replace quantification with Skolem function  $((\langle \text{type} \rangle \langle \text{var} \rangle \langle \text{type} \rangle \langle \text{expression} \rangle) \{ \langle \text{var} \rangle, \langle \text{var} \rangle \})$

Planning, Execution & Learning: Partial Order 12 Simmons : Spring 2007

## UCPOP & MTC

- The Modal Truth Criterion was used to prove that, for expressive operator representations, determining whether a plan achieves its conditions is NP-hard!
- UCPOP can handle expressive operators, yet it can trivially determine whether it has found a plan that achieves all the conditions
- How to reconcile this apparent contradiction?
  - MTC *proves whether*: Need to find necessary and sufficient conditions
  - UCPOP *ensures achievement*: Only need sufficient conditions
  - UCPOP pushes complexity from per-node cost to search space size
  - This is a **win** if search is (usually) well focused

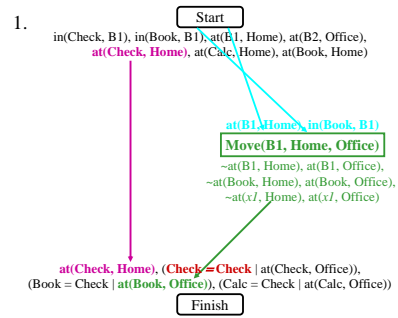
## UCPOP Algorithm

- UCPOP(initial-state, goals)
  - plan = <A={Start, Finish}, O={Start < Finish}, B={}, L={}>
  - agenda = {{goals, Finish}}
  - Repeat until agenda is empty
    - Select** (and remove) an *open condition* (q, a) from agenda
    - If q is quantified, then expand and add it to agenda
    - If q is a conjunction, then add each conjunct to agenda
    - If q is a disjunction, then **choose** one disjunct and add to agenda
    - If q is a literal and  $a_p \rightarrow a_e$  exists in L, then **Fail**
    - Else **choose**  $a_e$  (either a new action or an existing action from A) that has an effect r that unifies with q
      - Add  $\{a_e \rightarrow a_e\}$  to L
      - Add MGU(q, r, B) to B
      - Add  $\{(a_e < a_e), (a_e < Finish), (Start < a_e)\}$  to O
      - If  $a_e$  is new, add preconditions to agenda and any variable constraints to B
  - For each causal link  $a_e \rightarrow a_p$  and each  $a_e$  action which threatens the link, **choose** a resolution mechanism
    - Promotion*: Add  $\{a_e < a_p\}$  to O
    - Demotion*: Add  $\{a_e < a_p\}$  to O
    - Confrontation*: If threatening effect is conditional, with antecedent S and effect R, add  $\{(\sim\text{SMGU}(p, r, B), a_p)\}$  to agenda
- Fail** if plan is inconsistent

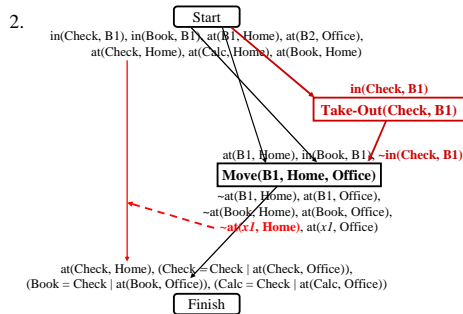
## UCPOP and the Briefcase World

- Move**(b, src, dest)
  - Pre: briefcase(b), at(b, src), src  $\neq$  dest
  - Effect: at(b, dest),  $\sim$ at(b, src),  
(forall (object x) (when in(x, b) (at(x, dest) &  $\sim$ at(x, src))))
- Take-Out**(x, b)      **Put-In**(x, b, loc)
  - Pre: in(x, b)              Pre: briefcase(b), at(x, loc), at(b, loc), x  $\neq$  b
  - Effect:  $\sim$ in(x, b)        Effect: in(x, b)
- Initial**: in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),  
at(Check, Home), at(Calc, Home), at(Book, Home),  
object(Check), object(Book), object(Calc),  
briefcase(B1), briefcase(B2)
- Goal**: at(Check, Home), (forall (object x) (x = Check | at(x, Office)))

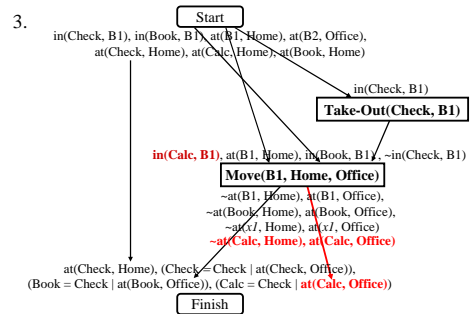
## UCPOP Briefcase World Example



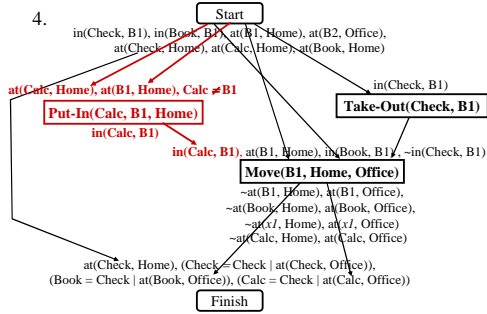
## UCPOP Briefcase World Example



## UCPOP Briefcase World Example



### UCPOP Briefcase World Example



### Partial Order Planning: Discussion

- **Advantages**

- Partial order planning is *sound* and *complete*
- Typically produces *optimal* solutions (plan length)
- Least commitment may lead to shorter search times

- **Disadvantages**

- Significantly more complex algorithms (higher *per-node* cost)
- Hard to determine what is true in a state
- Larger search space (**infinite!**)