Planning, Execution & Learning: Plan-Space Planning

Reid Simmons

Manuel Veloso

Where Does Planning Complexity Come From?

- Searching the Space
  - Worst case exponential (in length of plan)
- Per Node Costs
  - Generating successor nodes
  - Determining whether goal is met
  - Calculating heuristics
- Often Tradeoff
  - Doing more work at each node vs. doing more search

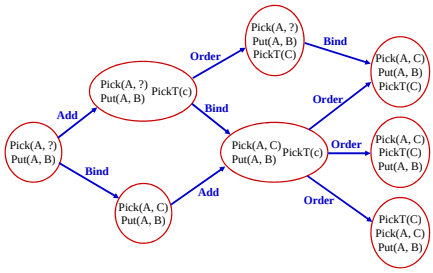
State-Space Search

- Search space is set of ***states***
- Search applies ***actions/operators*** to states to yield new states
- Plan is generated by the ***search path*** through the state space

Plan-Space Search

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

Plan-Space Search



Least Commitment

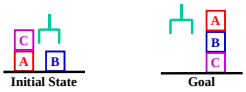
- Basic Idea
  - ***Make choices that are only relevant to solving the current part of the problem***
- Least Commitment Choices
  - **Orderings**: Leave actions unordered, unless they must be sequential
  - **Bindings**: Leave variables unbound, unless needed to unify with conditions being achieved
  - **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

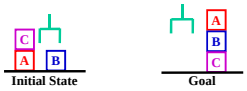
Plan-Space, Partial-Order Planning

- 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 blocks world problems that involve goal interactions (e.g., the "Sussman Anomaly" problem)

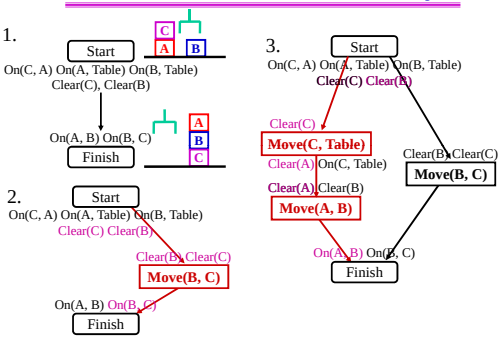


Sussman's Anomaly

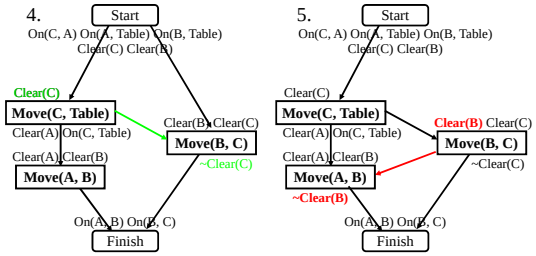
- Early linear (state-space) planners had trouble solving simple problems involving goal interactions (i.e., achieving one subgoal "clobbers" another)
  - Gerry Sussman (MIT) came up with a simple example of this
  - Partial-order planning can easily (and optimally) solve "Sussman's Anomaly," and similar problems



POP and Sussman's Anomaly



POP and Sussman's Anomaly



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_j
  - ***Demotion*** : Order a_k before a_i
  - ***Separation*** : Constrain c' so that it does not unify with c (non-codesignation constraint)

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: MGU(p, q, B) = {(v_i, x_i), … }
- Uses ***constraint satisfaction*** to prove consistency of plans
  - Consistent orderings
  - Consistent variable bindings (co-designation)

UCPOP Language Extensions

- Conditionals
  - *(when (?b ≠ table) (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: *(forall (<type> <var>) <expression>)*
  - ***Universal***: Expand into equivalent conjunct (assumes finite, known universe of objects)
  - ***Existential***: Replace quantification with Skolem function *((<type> <var$_i$>) & <expression>\{(<var>, <var$_i$>)})*

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_c) 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 $a_p \rightarrow^{\neg q} a_c$ exists in L, then **Fail**
    - Else **choose** a_p (either a new action or an existing action from A) that has an effect r that unifies with q
      - Add {$a_p \rightarrow^{q} a_c$ } to L
      - Add MGU(q, r, B) to B
      - Add {($a_p < a_c$), (a_p < *Finish*), (*Start* < a_p)} to O
      - If a_p is new, add preconditions to *agenda* and any variable constraints to B
    - For each causal link $a_i \rightarrow^{p} a_j$ and each a_t action which threatens the link, **choose** a resolution mechanism
      - ***Promotion***: Add ($a_j < a_t$) to O
      - ***Demotion*** : Add ($a_t < a_i$) to O
      - ***Confrontation*** : If threatening effect is conditional, with antecedent S and effect R, add {(~S\MGU(p, r, B), a_t)} to *agenda*
    - **Fail** if *plan* is inconsistent

Blue ***indicates non-deterministic selection among options***
Red ***indicates non-deterministic backtracking choice point***

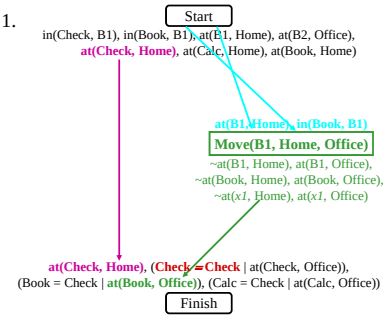
UCPOP and the Briefcase World

- **Move**(b, src, dest)
  Pre: briefcase(b), at(b, src), src ≠ dest
  Effect: at(b, dest), ~at(b, src),
  (forall (object x) (when in(x, b) (at(x, dest) & ~at(x, src))))
- **Take-Out**(x, b)
  Pre: in(x, b)
  Effect: ~in(x, b)

  Put-In(x, b, loc)
  Pre: briefcase(b), at(x, loc), at(b, loc), 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)))

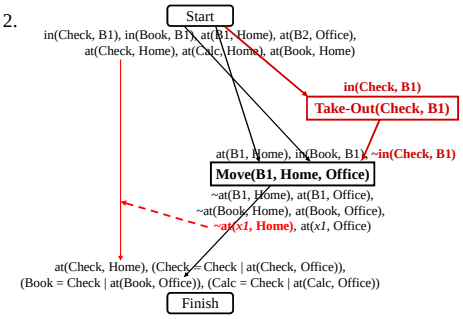
UCPOP Briefcase World Example

1.



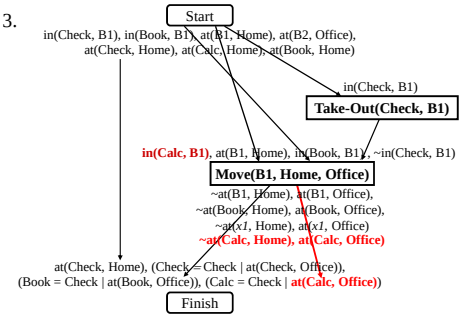
UCPOP Briefcase World Example

2.



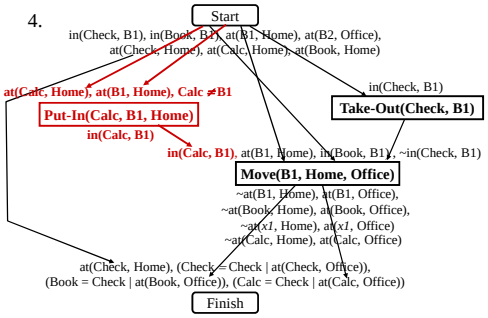
UCPOP Briefcase World Example

3.



UCPOP Briefcase World Example

4.



Partial Order Planning: Discussion

- **Advantages**
  - Partial order planning is ***sound*** and ***complete***
  - Typically produces better solutions (shorter), but not guaranteed ***optimal***
  - 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**!)