Partial-Order Planning

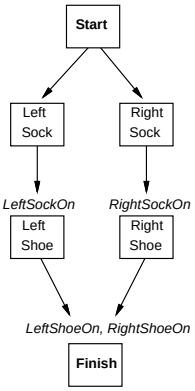
State-Space vs. Plan-Space

- **State-space** (**situation space**) planning algorithms search through the space of possible states of the world searching for a path that solves the problem.
- They can be based on **progression**: a forward search from the initial state looking for the goal state.
- Or they can be based on **regression**: a backward search from the goals towards the initial state
- STRIPS is an incomplete regression-based algorithm.
- **Plan-space** planners search through the space of partial plans, which are sets of actions that may not be totally ordered.
- **Partial-order** planners are plan-based and only introduce ordering constraints as necessary (**least committment**) in order to avoid unecessarily searching through the space of possible orderings.

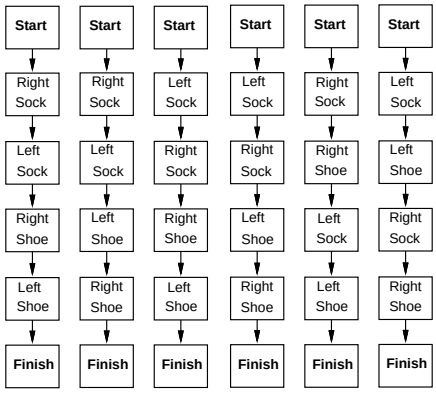
Partial Order Plans

- Plan in which not all actions are ordered

Partial Order Plan:



Total Order Plans:

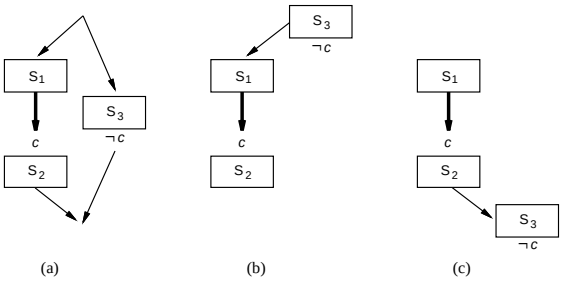


Plans, Causal Links, and Threats

- A *plan* is a three tuple <A, O, L>
  - A: A set of actions in the plan, $\{A_1,A_2,...A_n\}$
  - O: A set of ordering constraints on actions $\{A_i<A_j, A_k<A_l,...A_m<A_n\}$. These must be *consistent*, i.e. there must be at least one total ordering of actions in A that satisfy all the constraints.
  - B: A set of variable binding constraints {v=x,...}
  - L: a set of causal links showing how actions support each other
- A *causal link*, $A_p \rightarrow^Q A_c$, indicates that action A_p has an effect Q that achieves precondition Q for action A_c.
- A *threat*, is an action A_t that can render a causal link $A_p \rightarrow^Q A_c$ ineffective because:
  - $O \cup \{A_P < A_t < A_c\}$ is consistent
  - A_t has $\neg Q$ as an effect

Threat Removal

- Threats must be removed to prevent a plan from failing.
- **Demotion** adds the constraint $A_t < A_p$ to prevent clobbering, i.e. push the clobberer before the producer.
- **Promotion** adds the constraint $A_c < A_t$ to prevent clobbering, i.e. push the clobber after the consumer.



Initial (Null) Plan

- Initial plan has
  - A={ A_0, A_∞}
  - O={$A_0 < A_\infty$}
  - L ={}
- A_0(Start) has no preconditions but all facts in the initial state as effects.
- A_∞ (Finish) has the goal conditions as preconditions and no effects.

Op(Action: Go(there); Precond: At(here);
Effects: At(there), ¬At(here))
Op(Action: Buy(x), Precond: At(store), Sells(store,x);
Effects: Have(x))

POP Algorithm

- Stated as nondeterministic algorithm where choices must be made. Various search methods can be used (e.g. breadth-first, depth-first iterative deepening etc.) to explore the space of possible choices.
- Maintains agenda of goals that need to be supported by links, where an agenda element is a pair <Q,A_i> where Q is a precondition of A_i that needs supporting.
- Initialize plan to null plan and agenda to conjunction of goals (preconditions of Finish).
- Done when all preconditions of every action in plan are supported by causal links which are not threatened.

POP(<A,O,L>, agenda)

1) **Termination:** If agenda is empty, return <A,O,L>. Use topological sort to determine a totally ordered plan.

2) **Goal Selection:** Let <Q,A_{need}> be a pair on the agenda

3) **Action Selection:** Let A_{add} be a nondeterministically chosen action that adds Q. It can be an existing action in A or a new action. If there is no such action return failure.

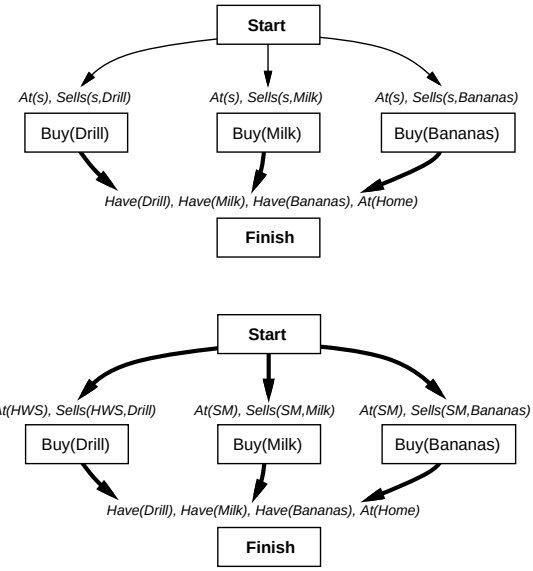
$L' = L \cup \{A_{add} \rightarrow^{Q} A_{need}\}$
$O' = O \cup \{A_{add} < A_{need}\}$
if A_{add} is new
then $A' = A \cup \{A_{add}\}$ and $O' = O' \cup \{A_0 < A_{add} < A_\infty\}$
else $A' = A$

4) **Update goal set:** Let agenda´= agenda – {<Q,A_{need}>}
If A_{add} is new then for each conjunct Q_i of its precondition, add <Q_i, A_{add}> to agenda´

5) **Causal link protection:** For every action A_t that threatens a causal link $A_p \rightarrow^{Q} A_c$ add an ordering constraint by choosing nondeterministically either
(a) **Demotion:** Add $A_t < A_p$ to O´
(b) **Promotion:** Add $A_c < A_t$ to O´
If neither constraint is consistent then return failure.

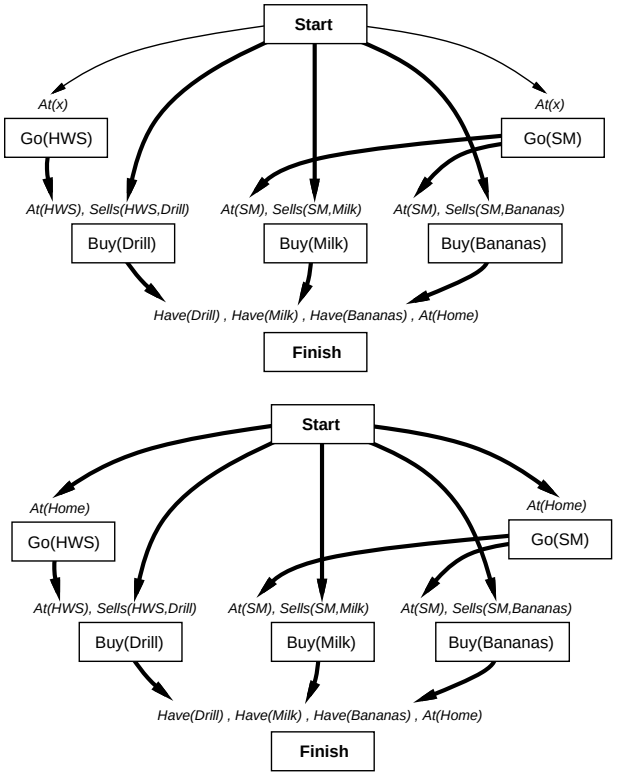
6) **Recurse:** POP(<A´,O´,L´>, agenda´)

Example

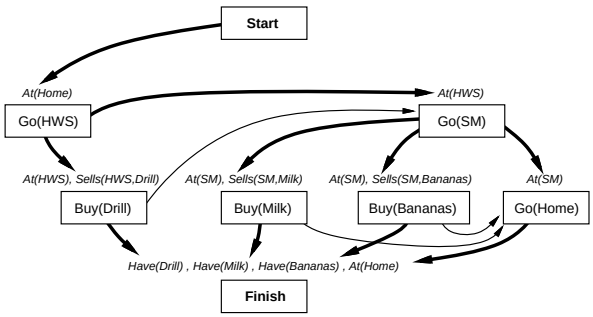


- Add three actions to achieve basic goals. Use initial state to achieve the Sells preconditions.
- Bold links are causal, regular are just ordering constraints

Example (cont)

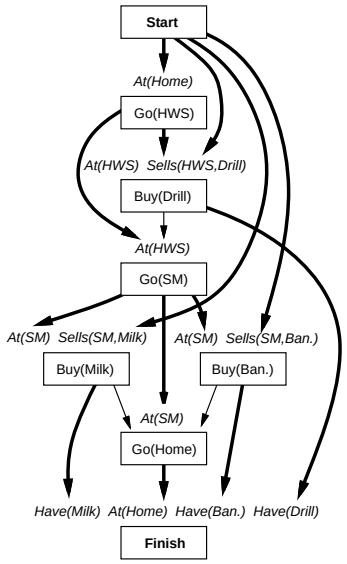


Example (cont)



- Cannot resolve threat to At(Home) preconditions of both Go(HWS) and Go(SM).
- Must backtrack to supporting At(x) precondition of Go(SM) from intial state At(Home) and support it instead from the At(HWS) effect of Go(HWS).
- Since Go(SM) still threatens At(HWS) of Buy(Drill) must promote Go(SM) to come after Buy(Drill). Demotion is not possible due to causal link supporting At(HWS) precondition of Go(SM)

Example (cont)



- Add Go(Home) action to achieve At(Home), use At(SM) to achieve its precondition, and order it after Buy(Milk) and Buy(Banana) to resolve threats to At(SM).

Planning Conclusions

- Experiments confirm that in most cases partial-order planning is more efficient than total order.

- Planning techniques have been applied to a number of realistic tasks:
  - Logistics planning for Desert Storm
  - Scheduling for the Hubble Space Telescope
  - Planning ground operations for the Space Shuttle
  - Semiconductor manufacturing
  - Cleaning oil spills

- Many issues are involved in interleaving planning and execution
  - Conditional planning
  - Execution monitoring and dynamic replanning