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 commitment) in order to avoid unnecessarily searching through the space of possible orderings.

Partial Order Plans

- Plan in which not all actions are ordered

\begin{itemize}
  \item Plan in which not all actions are ordered
  \item A plan is a three tuple \( <A, O, L> \)
  \item \(-A\): A set of actions in the plan, \( \{A_1, A_2, \ldots, A_n\} \)
  \item \(-O\): A set of ordering constraints on actions \( \{A_i < A_j, A_k < A_l, \ldots, A_m < A_n\} \). These must be consistent, i.e. there must be at least one total ordering of actions in \( A \) that satisfy all the constraints.
  \item \(-B\): A set of variable binding constraints \( \{v=x, \ldots\} \)
  \item \(-L\): a set of causal links showing how actions support each other
  \item A causal link, \( A_p \rightarrow^O A_c \), indicates that action \( A_p \) has an effect \( Q \) that achieves precondition \( Q \) for action \( A_c \).
  \item A threat, is an action \( A_t \) that can render a causal link \( A_p \rightarrow^O A_c \) ineffective because:
    \begin{itemize}
      \item \(-O \cup (A_p < A_i < A_c)\) is consistent
      \item \(-A_i\) has \(-Q\) as an effect
    \end{itemize}
\end{itemize}
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 clobber before the producer.
- Promotion adds the constraint $A_c < A_t$ to prevent clobbering, i.e., push the clobber after the consumer.

![Diagram]

Initial (Null) Plan

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

Op (Action: Go(there); Precond: At(there); Effects: At(there), $\neg$At(there))
Op (Action: Buy(x), Precond: At(store), Sells(store,x); Effects: Have(x))

![Diagram]

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 $<Q,A_{\text{need}}>\text{ where } Q\text{ is a precondition of } A_t$ 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.

![Diagram]

POP($<A,O,L>, \text{agenda}$)

1) **Termination**: If agenda is empty, return $<A,O,L>$. Use topological sort to determine a totally ordered plan.
2) **Goal Selection**: Let $<Q,A_{\text{need}}>\text{ be a pair on the agenda}$
3) **Action Selection**: Let $A_{\text{add}}$ 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.
   - $L' = L \cup \{ A_{\text{add}} \rightarrow Q | A_{\text{need}} \}$
   - $O' = O \cup \{ A_{\text{add}} < A_{\text{need}} \}$
   - If $A_{\text{add}}$ is new then $A' = A \cup \{ A_{\text{add}} \}$ and $O' = O' \cup \{ A_0 < A_{\text{add}} < A_\infty \}$
   - Else $A' = A$
4) **Update goal set**: Let agenda$' =\text{agenda} - \{ <Q,A_{\text{need}}> \}$
   - If $A_{\text{add}}$ is new then for each conjunct $Q_i$ of its precondition, add $<Q_i,A_{\text{add}}>$ to agenda$'$
5) **Causal link protection**: For every action $A_t$ that threatens a causal link $A_p \rightarrow Q | A_0$, 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($<A',O',L'>, \text{agenda}')$
Example

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

Example (cont)

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

Example (cont)

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