Backward State Space Search Planning
(Regression Planning)

Idea: Want to generate possible predecessors of a given goal state, work backwards toward the initial state.

Hope is this way only look at relevant actions.

How to go backward? given a state S and an action A

1. Delete positive effects of A from s'
2. Add any precondition literal of A to S' unless it already appears.

Result is predecessor state S.

Example: Action(ApplyFingerNose, Precondition: NoseClear, Effect NosePlugged)

State S' was NosePlugged then S is NoseClear

Note: In either forward or backward search, there might be several states to choose from as the state to consider next.

Can make up heuristic functions which estimate how close we are to a solution

Example: # of literals not yet filled.
Using this heuristic, can use A* to choose next state to consider.

Above planners are so called total order planners.

They are called total order planners because they produce only one possible fixed plan.

Partial Order Planning –
Idea – Want to be able to decompose problems into subtasks, produce plans for each subtask. Choose subtasks so doesn't matter the order that we do them.

Example: Want to put both shoes on. Expressing this in a strips style situation calculus.
Goal(RightShoeon AND LeftShoeon)
Init()
Action(RightShoe, PreCondition: RightSockOn, Effect: RightShoeon)
Action(RightSock, Effect: RightSockOn)

Similar actions for left foot

Can create two subplans one to put each sock on.

Init -> RightSock -> RightShoe -> RightShoeOn
Init -> LeftSock -> LeftShoe -> LeftShoeOn

Partial Order Plan (POP)

$$\begin{tikzcd}
\text{Init} \arrow[r] \arrow[dr] & \text{PlanL} \arrow[r] & \text{Goal} \\
& \text{PlanR} \arrow[ur] &
\end{tikzcd}$$

To execute we choose some total ordering
Init -> PlanL -> PlanR -> Goal
Init -> PlanR -> PlanL -> Goal

To allow concurrency, we could also interleave operations from the two plans