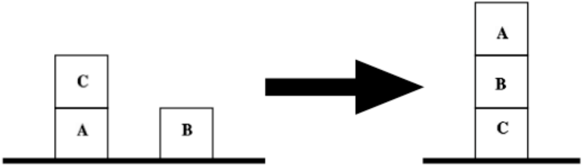
Automated (AI) Planning

Planning as Plan-Space Search

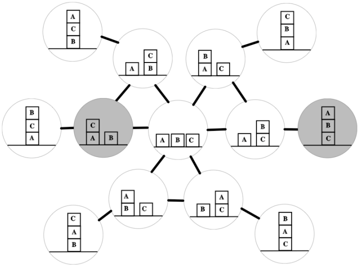
Carmel Domshlak

State Space Search

- So far we have considered planning as search in state space
  - forward - build a plan in the same order that it is executed
  - backward - build a plan in the reverse order of its execution
  - temporal undirected - unordered commitments on executing actions in time



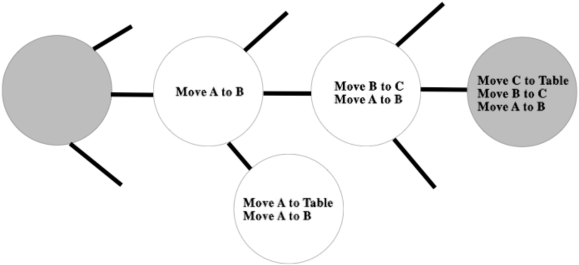
State Space Search



- Potential problem:
  Spending lots of time on trying the same set of actions in different orderings before realizing that there is no solution (with this set)
  - Easier to see in FS/BS, and a bit harder to see in TUS.
- Key observation: When we choose what to do, we also choose when to do

Searching in the Space of Plans

- In 1974, Earl Sacerdoti built a planner, called *NOAH*, that considered planning as search through plan space
  - Search states (nodes) = partially specified plans
  - Transitions (edges) = plan refinement operations
  - Initial state = null plan
  - Goal states = valid plans for the problems



State Space vs. Plan Space

- Search through plan space ... hmm ... *what is plan*?

- Answer I: Totally ordered sequence of either actions or meta-actions
  - But then search through state space is isomorphic to search through plan space!
  - Hmmm . . . the nature of the space being searched is in the eye of the beholder ...
  - So what is the point of introducing "search through plan space"??

- Answer II: Partially ordered sequence of actions

State Space vs. Plan Space

- Search through plan space ... hmm ... *what is plan*?
- Answer I: Totally ordered sequence of either actions or meta-actions
  - But then search through state space is isomorphic to search through plan space!
  - Hmmm . . . the nature of the space being searched is in the eye of the beholder ...
  - So what is the point of introducing "search through plan space"??
- Answer II: Partially ordered sequence of actions

Least Commitment Planning

Think how *you* might solve a planning problem of ...
going for a vacation to Italy

1. Need to purchase plane tickets
2. Need to buy a "Lonely Planet" guide to Italy

BUT there is no need to decide (*yet*) which purchase should be done first

Least Commitment Planning

Think how *you* might solve a planning problem of ...
going for a vacation to Italy

1. Need to purchase plane tickets
2. Need to buy a “Lonely Planet” guide to Italy

BUT there is no need to decide (*yet*) which purchase should be done first

Least Commitment Planning

- Represent plans in a flexible way that enables deferring decisions
- At the planning phase, only the essential ordering decisions are recorded

Partial-Order Plans

- Given a Strips task $\Pi = (P, A, I, G)$ we search through a space of *hypothetical* partial-order plans
- A plan (= search node) is a triplet: $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$ in which
  - $\mathcal{A}$ is a set of actions from A, possibly with (labeled) repetitions
  - $\mathcal{O}$ is a set of ordering constraints over $\mathcal{A}$
  - $\mathcal{L}$ is a set of causal links (a bit later)
- Example: $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{O} = \{a_1 < a_3, a_2 < a_3\}$
- Observe: Planner (eventually) must do constraint satisfaction to ensure the consistency of $\mathcal{O}$.

Causal Links

A key aspect of least commitment planning is to keep track of past decisions and the *reasons* for those decisions

- If you purchase plane tickets, then make sure bring them to the airport
- If another goal causes you to drop the tickets (e.g., having you hands free to open the taxi door), then you should be sure to pick them up again.

- A good way to reason about (and act for) non-interference between different actions introduced to the plan is to record dependencies between actions *explicitly*
- Causal links $a_p \xrightarrow{q} a_c$ records our decision to use a_p to produce the precondition q of a_c

Causal Links

A key aspect of least commitment planning is to keep track of past decisions and the *reasons* for those decisions

- If you purchase plane tickets, then make sure bring them to the airport
- If another goal causes you to drop the tickets (e.g., having you hands free to open the taxi door), then you should be sure to pick them up again.

- A good way to reason about (and act for) non-interference between different actions introduced to the plan is to record dependencies between actions *explicitly*
- Causal links $a_p \xrightarrow{q} a_c$ records our decision to use a_p to produce the precondition q of a_c

Threats

- Causal links are used to detect when a newly introduced action interferes with past decisions.
- Such an action is called a threat

- Suppose that
  - $a_p \xrightarrow{q} a_c$ is a causal link in $\mathcal{L}$ (of some plan $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$), and
  - a_t is yet another action in $\mathcal{A}$
- We say that a_t threatens $a_p \xrightarrow{q} a_c$ if
  - $\mathcal{O} \cup \{a_p < a_t < a_c\}$ is consistent, and
  - $q \in \mathsf{del}(a_t)$

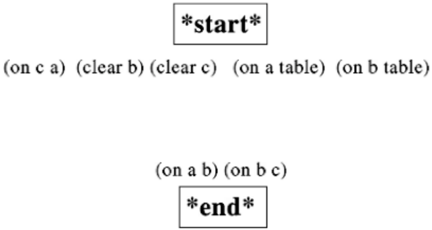
Eliminating Threats

- When a plan contains a threat, then it is *possible* that the plan would not work as anticipated.
  - *Which means what?*
- Solution: identify threats and take evasive countermeasures
  - promotion by $\mathcal{O} \cup= \{a_t > a_c\}$
  - demotion by $\mathcal{O} \cup= \{a_t < a_p\}$
  - ...

Planning Problems as Null Plans

Uniformity is the key for simplicity

- Can use the same structure to represent both the planning problem and complete plans
- Planning problem as a **null plan** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$ where
  - $\mathcal{A} = \{a_0, a_\infty\}$, $\mathcal{O} = \{a_0 < a_\infty\}$, $\mathcal{L} = \{\}$
  - $\text{pre}(a_0) = \{\}, \text{del}(a_0) = \{\}, \text{add}(a_0) = I$
  - $\text{pre}(a_\infty) = G, \text{del}(a_0) = \{\}, \text{add}(a_0) = \{\}$



The POP Algorithm
Schematic description

Regressive algorithm that searches plan-space

- Starts with the null plan
- Makes *non-deterministic* plan refinement choices until
  - all preconditions of all actions in the plan have been supported by causal links, and
  - all threatened causal links have been protected from possible interference

The POP Algorithm
Input and Output

Recursive calls to POP with $POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$ where

- $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$ is a plan structure
- $agenda$ is a list of "open goals" that need to be supported by causal links
- A is the action set of our Strips problem

Initial call is with

- null plan $\langle \{a_0, a_\infty\}, \{a_0 < a_\infty\}, \{\} \rangle$, and
- $agenda = \{(g, a_\infty) \mid g \in \mathsf{pre}(a_\infty) \equiv G\}$

If $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$ is outputted by POP, then *any* total ordering of actions $\mathcal{A}$ consistent with $\mathcal{O}$ is a valid plan for our problem.

The POP Algorithm

$POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

- Termination: **if** $agenda = \emptyset$ **then return** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
- Goal selection: **choose** $(q, a_{need}) \in agenda$

The POP Algorithm

$POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

- Termination: **if** $agenda = \emptyset$ **then return** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
- Goal selection: **choose** $(q, a_{need}) \in agenda$
- Action selection:
  - **choose** action a_{add} (either from $\mathcal{A}$, or from A) such that
    - $q \in$ add(a_{add}), and
    - $\mathcal{O} \cup \{a_{add} < a_{need}\}$ is consistent
  - **if** no such action **then return** FALSE
  - **otherwise**
    - $\mathcal{L} \cup= \{a_{add} \xrightarrow{q} a_{need}\}$ and $\mathcal{O} \cup= \{a_{add} < a_{need}\}$
    - **if** a_{add} is a new action instance **then** $\mathcal{A} \cup= \{a_{add}\}$, and $\mathcal{O} \cup= \{a_0 < a_{add} < a_\infty\}$

The POP Algorithm

$POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

- Termination: **if** $agenda = \emptyset$ **then return** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
- Goal selection: **choose** $(q, a_{need}) \in agenda$
- Action selection:
  - **choose** action a_{add} (either from $\mathcal{A}$, or from A) such that
    - $q \in \text{add}(a_{add})$, and
    - $\mathcal{O} \cup \{a_{add} < a_{need}\}$ is consistent
  - **if** no such action **then return** FALSE
  - **otherwise**
    - $\mathcal{L} \cup= \{a_{add} \xrightarrow{q} a_{need}\}$ and $\mathcal{O} \cup= \{a_{add} < a_{need}\}$
    - **if** a_{add} is a new action instance **then** $\mathcal{A} \cup= \{a_{add}\}$, and $\mathcal{O} \cup= \{a_0 < a_{add} < a_\infty\}$
- Update goal set:
  - $agenda \setminus= \{(q, a_{need})\}$
  - **if** a_{add} was a new action instance **then** $agenda \cup= \{(r, a_{add}) \mid r \in \text{pre}(a_{add})\}$

The POP Algorithm

$POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

- Termination: **if** $agenda = \emptyset$ **then return** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
- Goal selection: **choose** $(q, a_{need}) \in agenda$
- Action selection: **choose** and **process** a_{add} ...
- Update goal set: add preconditions of a_{add} to the agenda

  ...

- Causal link protection: **foreach** causal link $\{a_p \xrightarrow{r} a_c\} \in \mathcal{L}$, and a_t that is threatening it
  - **choose** either $\mathcal{O} \cup= \{a_t > a_c\}$, or $\mathcal{O} \cup= \{a_t < a_p\}$
  - **if** neither constraint is consistent **then return** FALSE

The POP Algorithm

$POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

- Termination: **if** $agenda = \emptyset$ **then return** $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
- Goal selection: **choose** $(q, a_{need}) \in agenda$
- Action selection: **choose** and **process** a_{add} ...
- Update goal set: add preconditions of a_{add} to the agenda

  ...

- Causal link protection: **foreach** causal link $\{a_p \xrightarrow{r} a_c\} \in \mathcal{L}$, and a_t that is threatening it
  - **choose** either $\mathcal{O} \cup= \{a_t > a_c\}$, or $\mathcal{O} \cup= \{a_t < a_p\}$
  - **if** neither constraint is consistent **then return** FALSE
- Recursive invocation: $POP(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, agenda, A)$

Choice Points

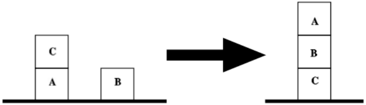
Three choice points

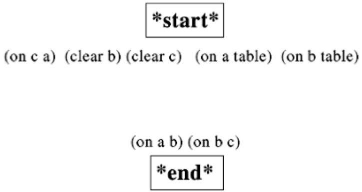
- Goal selection
- Action selection
- Causal link protection

How crucial these choices are?

- Affect soundness?
- Affect completeness?
- Affect efficiency?

Example - Step 1





Initial call to POP with

- Null Plan (see the right figure)
- $agenda = \{(\texttt{onAB}, a_\infty), (\texttt{onBC}, a_\infty)\}$

First choice is *goal selection*

- Affects efficiency, but *not* completeness!

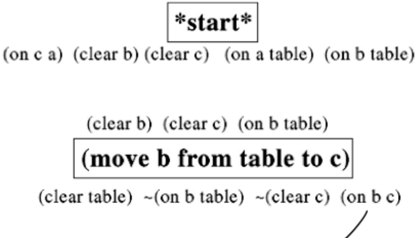
Example - Step 2

Suppose $(\texttt{onBC}, a_\infty)$ is selected (i.e., $a_{need} = a_\infty$)

- Need to choose an action a_{add} that will provide onBC
  - This is a real non-deterministic choice!

Suppose that an oracle suggests making a_{add} be a new instance of the action *move-B-from-Table-to-C*

- a causal link $a_{add} \xrightarrow{\texttt{onBC}} a_\infty$ is added to $\mathcal{L}$
- *agenda* is properly updated *(how exactly?)*
- no threats to resolve . . . recursive call



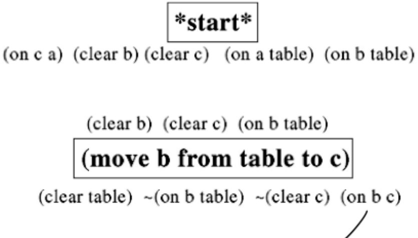
Example - Step 2

Suppose $(\texttt{onBC}, a_\infty)$ is selected (i.e., $a_{need} = a_\infty$)

- Need to choose an action a_{add} that will provide onBC
  - This is a real non-deterministic choice!

Suppose that an oracle suggests making a_{add} be a new instance of the action $move\text{-}B\text{-}from\text{-}Table\text{-}to\text{-}C$

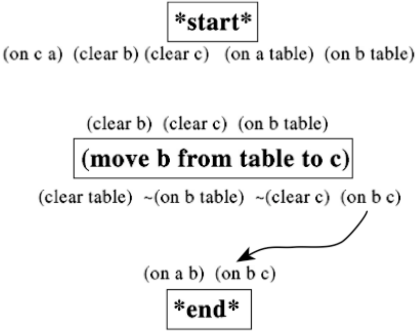
- a causal link $a_{add} \xrightarrow{\texttt{onBC}} a_\infty$ is added to $\mathcal{L}$
- $agenda$ is properly updated *(how exactly?)*
- no threats to resolve . . . recursive call



Example - Step 2

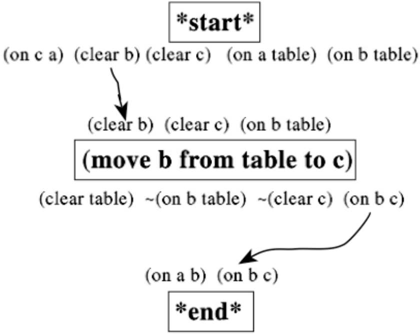
Suppose that an oracle suggests making a_{add} be a new instance of the action $move\text{-}B\text{-}from\text{-}Table\text{-}to\text{-}C$

- a causal link $a_{add} \xrightarrow{\texttt{onBC}} a_{\infty}$ is added to $\mathcal{L}$
- $agenda$ is properly updated *(how exactly?)*
- no threats to resolve . . . recursive call



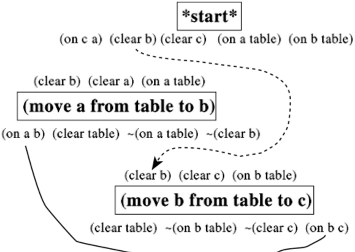
Example - Step 3

- Suppose $(\texttt{clearB}, \textit{move-B-from-Table-to-C})$ is selected
- Oracle suggests to reuse an existing action instance a_0
  - add a causal link $a_0 \xrightarrow{\texttt{clearB}} \textit{move-B-from-Table-to-C}$
  - $\textit{agenda}$ is properly updated *(how exactly?)*
  - no threats to resolve . . . recursive call



Example - Step 4a

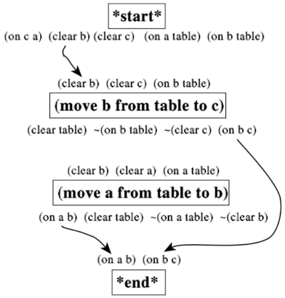
- Suppose $(\texttt{onAB}, a_\infty)$ is selected
- Oracle suggests making a_{add} be a new instance of the action $move\text{-}A\text{-}from\text{-}Table\text{-}to\text{-}B$, and we do that ...
- ... BUT this time we have a threat!
  - $move\text{-}A\text{-}from\text{-}Table\text{-}to\text{-}B$ and $move\text{-}B\text{-}from\text{-}Table\text{-}to\text{-}C$ have no constraints on their relative ordering
  - $move\text{-}A\text{-}from\text{-}Table\text{-}to\text{-}B$ deletes `clearB` that is required by $move\text{-}B\text{-}from\text{-}Table\text{-}to\text{-}C$



Example - Step 4b

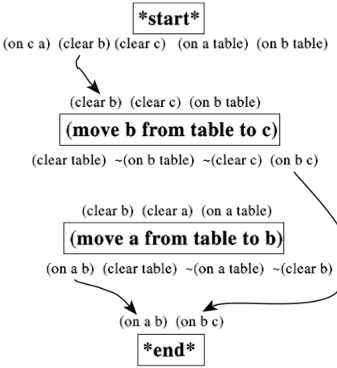
Try to protect the causal link
$a_0 \xrightarrow{\texttt{clearB}} move\text{-}B\text{-}from\text{-}Table\text{-}to\text{-}C$

- In general, there are two options — *promotion* and *demotion* — and this is a true non-deterministic choice!
- In our example, demotion is inconsistent *(why?)*, but promotion is OK



Example - Next steps

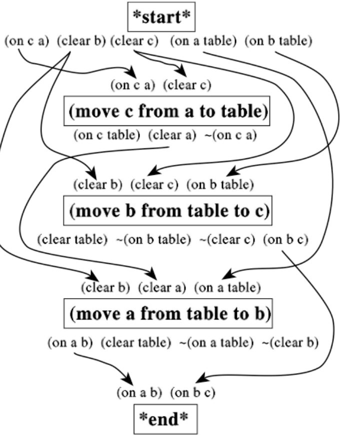
- What is now on the $agenda$? ...in $\mathcal{A}$? ...in $\mathcal{L}$? ...in $\mathcal{O}$?



- Next steps follow the same lines of reasoning

Example - Next steps

- Eventually POP returns



- Blackboard: *Is it a correct partial order plan?*

Advantages

- Natural extension to planning with partially instantiated actions
  - ... add action instance $move\text{-}A\text{-}from\text{-}x?\text{-}to\text{-}B$
  - ... postpone unifying $?x$ with a concrete object until necessary
- Natural extensions to more complex action formalisms
  - ... action durations
  - ... delayed effects
  - ...
- Least commitment may lead to shorter search times
  - Mainly due to smaller branching factor

Disadvantages

- Significantly more complex algorithm
  - ... higher *per-node* cost
- Hard to determine what is true in a state
  - ... harder to devise informed heuristics (for all three types of choices)
  - ... how to prune infinitely long paths??