Planning, Execution & Learning
1. Partial Order Planning

Reid Simmons
Partial Order Planning

• Basic Idea
  – *Search in plan space and use least commitment, when possible*

• 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
**Least Commitment**

- Basic Idea
  - *Make choices only that are 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 $a_i$
**NOAH [Sacerdoti, 1975]**

- NOAH
  - First non-linear, partial-order planner
  - Introduced notion of plan-space search
  - Used *TOME* (*Table of Multiple Effects*) to detect goal interactions

- NOAH can easily (and optimally) solve the “Sussman Anomaly” problem
**NOAH and Sussman’s Anomaly**

1. Start
   On(C, A) On(A, Table) On(B, Table)
   Clear(C), Clear(B)
   On(A, B) On(B, C)
   Finish

2. Start
   On(C, A) On(A, Table) On(B, Table)
   Clear(C) Clear(B)
   Clear(B) Clear(C)
   Move(B, C)
   On(A, B) On(B, C)
   Finish

3. Start
   On(C, A) On(A, Table) On(B, Table)
   Clear(C) Clear(B)
   Move(C, Table)
   Clear(A) On(C, Table)
   Move(B, C)
   Clear(B) Clear(C)
   Move(A, B)
   On(A, B) On(C, Table)
   Move(B, C)
   Clear(B) Clear(C)
   Finish
NOAH and Sussman’s Anomaly

4. 
Start
On(C, A) On(A, Table) On(B, Table)
Clear(C) Clear(B)

\[ \text{Move}(A, B) \]
Clear(A) On(C, Table)
Clear(A) Clear(B)

\[ \text{Move}(A, B) \]
On(A, B) On(B, C)

\[ \text{Finish} \]

\[ \neg \text{Clear}(C) \]

\[ \text{Move}(B, C) \]
Clear(B) Clear(C)

5. 
Start
On(C, A) On(A, Table) On(B, Table)
Clear(C) Clear(B)

\[ \text{Move}(C, Table) \]
Clear(A) On(C, Table)
Clear(A) Clear(B)

\[ \text{Move}(A, B) \]
On(A, B) On(B, C)

\[ \neg \text{Clear}(C) \]

\[ \text{Move}(B, C) \]
Clear(B) Clear(C)

\[ \text{Move}(A, B) \]
On(A, B) On(B, C)

\[ \neg \text{Clear}(B) \]

\[ \text{Finish} \]
Modal Truth Criterion [Chapman, 1987]

- Modal Truth Criterion (MTC)
  - Formalized criterion for determining whether a (partial) plan achieves a given precondition \( p \) at a given step \( s \)
  - \( p \) is true in \( s \) if:
    \[
    \exists t \ ( (t < s) \land \text{asserts}(t, p)) \land \\
    \forall C \ ( (s < C) \lor \\
    \forall q \ ((\Diamond q \approx p) \Rightarrow \neg\text{denies}(C, q)) \lor \\
    \exists W \ ( (C < W) \land (W < s) \land \\
    \exists r (\text{asserts}(W, r) \land (p \approx q) \Rightarrow (p \approx r))))
    \]

- Can be used to generate planning algorithm (TWEAK)
  - step addition / establishment
  - promotion/demotion
  - separation
  - white knight
**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 = \( <A, O, B, L> \)

- 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: $\text{MGU}(p, q, B) = \{(v_i, x_i), \ldots\}$

- 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>)\})
The Modal Truth Criterion was used to prove that, for expressive operator representations, determining whether a plan achieves its conditions is NP-hard!

UCPOP can handle expressive operators, yet it can trivially determine whether it has found a plan that achieves all the conditions

How to reconcile this apparent contradiction?

- MTC *proves whether*: Need to find necessary and sufficient conditions
- UCPOP *ensures achievement*: Only need sufficient conditions
- UCPOP pushes complexity from per-node cost to search space size
- This is a *win* if search is (usually) well focused
UCPOP Algorithm

- UCPOP(initial-state, goals)
  - plan = \langle A\{\text{Start}, \text{Finish}\}, O\{\text{Start < Finish}\}, B\{\}, L\{\} \rangle
  - agenda = \{(goals, \text{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 \(\text{MGU}(q, r, B)\) to \(B\)
      - Add \(\{(a_p < a_c), (a_p < \text{Finish}), (\text{Start} < a_p)\}\) 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 \((a_j < a_i)\) to \(O\)
      - Demotion: Add \((a_i < a_j)\) to \(O\)
      - Confrontation: If threatening effect is conditional, with antecedent \(S\) and effect \(R\), add \(\{\neg S \text{\text{\textbackslash MGU}}(p, r, B), a_t\}\) to agenda
  - Fail if plan is inconsistent
UCPOP and the Briefcase World

- **Move** \((b, \text{src}, \text{dest})\)
  
  **Pre:** briefcase\((b)\), at\((b, \text{src})\), \(\text{src} \neq \text{dest}\)

  **Effect:** at\((b, \text{dest})\), \neg\text{at}(b, \text{src})

  \((\forall \text{object } x)(\text{when } \text{in}(x, b) (\text{at}(x, \text{dest}) \& \neg\text{at}(x, \text{src})))\))

- **Take-Out** \((x, b)\)    **Put-In** \((x, b, \text{loc})\)

  **Pre:** in\((x, b)\)    **Pre:** briefcase\((b)\), at\((x, \text{loc})\), at\((b, \text{loc})\), \(x \neq b\)

  **Effect:** \neg\text{in}(x, b)    **Effect:** in\((x, b)\)

- **Initial:** in\((\text{Check}, B1)\), in\((\text{Book}, B1)\), at\((B1, \text{Home})\), at\((B2, \text{Office})\), at\((\text{Check}, \text{Home})\), at\((\text{Calc}, \text{Home})\), at\((\text{Book}, \text{Home})\), object\((\text{Check})\), object\((\text{Book})\), object\((\text{Calc})\), briefcase\((B1)\), briefcase\((B2)\)

- **Goal:** at\((\text{Check}, \text{Home})\), \((\forall \text{object } x)(x = \text{Check} \mid \text{at}(x, \text{Office})))\)
1.

\begin{align*}
\text{in}(\text{Check}, \text{B1}), \text{in}(\text{Book}, \text{B1}), \text{at}(\text{B1}, \text{Home}), \text{at}(\text{B2}, \text{Office}), \\
\text{at}(\text{Check}, \text{Home}), \text{at}(\text{Calc}, \text{Home}), \text{at}(\text{Book}, \text{Home})
\end{align*}

\begin{align*}
\text{Move}(\text{B1}, \text{Home}, \text{Office}) \\
\text{at}(\text{B1}, \text{Home}), \text{in}(\text{Book}, \text{B1})
\end{align*}

\begin{align*}
\text{at}(\text{Check}, \text{Home}), (\text{Check} = \text{Check} | \text{at}(\text{Check}, \text{Office})), \\
(\text{Book} = \text{Check} | \text{at}(\text{Book}, \text{Office})), (\text{Calc} = \text{Check} | \text{at}(\text{Check}, \text{Office}))
\end{align*}

\begin{align*}
\text{Finish}
\end{align*}
2. Start

\begin{align*}
&\text{in}(\text{Check}, B1), \text{in}(\text{Book}, B1), \text{at}(B1, \text{Home}), \text{at}(B2, \text{Office}), \\
&\text{at}(\text{Check}, \text{Home}), \text{at}(\text{Calc}, \text{Home}), \text{at}(\text{Book}, \text{Home})
\end{align*}

Finish

\begin{align*}
&\text{at}(\text{Check}, \text{Home}), (\text{Check} = \text{Check} \mid \text{at}(\text{Check}, \text{Office})), \\
&(\text{Book} = \text{Check} \mid \text{at}(\text{Book}, \text{Office})), (\text{Calc} = \text{Check} \mid \text{at}(\text{Check}, \text{Office}))
\end{align*}
3. in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office),
at(Check, Home), at(Calc, Home), at(Book, Home)

Start

in(Check, B1)

Take-Out(Check, B1)

in(Calc, ?b), at(B1, Home), in(Book, B1), ~in(Check, B1)

Move(B1, Home, Office)

~at(B1, Home), at(B1, Office),
~at(Book, Home), at(Book, Office),
~at(x1, Home), at(x1, Office)
~at(Calc, Home), at(Calc, Office)

at(Check, Home), (Check = Check | at(Check, Office)),
(Book = Check | at(Book, Office)), (Calc = Check | at(Check, Office))

Finish
4. Start

\[ \text{in(Check, B1), in(Book, B1), at(B1, Home), at(B2, Office), at(Check, Home), at(Calc, Home), at(Book, Home)} \]

Put-In(Calc, B1, Home)

\[ \text{in(Calc, B1)} \]

\[ \text{in(Calc, B1), at(B1, Home), in(Book, B1), \sim\text{in(Check, B1)}} \]

Move(B1, Home, Office)

\[ \text{\sim at(B1, Home), at(B1, Office), \sim at(Book, Home), at(Book, Office), \sim at(x1, Home), at(x1, Office), \sim at(Calc, Home), at(Calc, Office)} \]

\[ \text{at(Check, Home), (Check = Check \mid at(Check, Office)), (Book = Check \mid at(Book, Office)), (Calc = Check \mid at(Check, Office))} \]

Finish
Partial Order Planning: Discussion

- **Advantages**
  - Partial order planning is *sound* and *complete*
  - Typically produces *optimal* solutions (plan length)
  - 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, since concurrent actions are allowed