Planning

- Idea: Use special purpose representations and algorithms for more efficient plan production.
  - Representation for situations, actions, goals, plans
  - Algorithms for searching
    - Situation space
    - Plan space

Planning and Acting

- Want algorithms for
  - Making a plan
  - Carrying out the plan
  - Monitoring the execution
    - Why? Plan might not work as expected; world might change
  - Learning from experience

The `STRIPS' Approach

- Represent a situation as knowledge base of (restricted) logical sentences.
- **Represent operators by how they should change the KB.**
- Reason about the implications of actions by changing the KB.
  - reversibly, so we can back-track cheaply.
Representation in STRIPS

- Situations
  - Conjunctions of ground literals (no functions)
    - Systems differ on whether this is supposed to be a complete or partial representation of the world
  - E.g.: \texttt{At(Box1,B) \land At(Box2,C)}
  - Of course, we still have general facts about the world that might be represented using implication, etc., but these won’t change from situation to situation.

- Goals
  - Same thing, but allow variables (interpreted existentially)
  - E.g., \texttt{At(Box1,x) \land At(Box2,x)}

STRIPS Operator (Schema) Example

- Go to one location from another.
  - Action Description: \texttt{Goto(m,n)}
  - Add: \texttt{At(Robot,n)}
  - Del: \texttt{At(Robot,m)}
  - Pre: \texttt{At(Robot,m)}

- Pushing a box from one location to another.
  - Action Description: \texttt{Push(k,m,n)}
  - Add: \texttt{At(Robot,n) \land At(k,n)}
  - Del: \texttt{At(Robot,m) \land At(k,m)}
  - Pre: \texttt{At(Robot,m) \land At(k,m) \land Box(k)}

- In general, preconditions might contain more information.

Using Operators to Search Situation Space

- Search the \textit{space of situations}, which is connected by operator instances, for a situation, accessible from the initial situation, in which the goal pertains.
- The sequence of operator instances is the plan.
- Simulate applying an operator instance by changing the knowledge base.
- In principle, one could use any number of different search strategies to find a plan.
  - Forward from initial state
  - Backward from goal state
  - Other

Representing Operators in STRIPS

- Action or Operator Schemata
  - Components
    - Action description: A way to name the action
    - \textbf{Add list}: What becomes true after the action
    - \textbf{Delete list}: What ceases to become true after the action
      - Add and Delete lists are sometimes combined into a single \textbf{Effects} list.
    - \textbf{Precondition}: What must be true to undertake the action.
    - We can only execute actions, meaning a fully instantiated schema.
(What STRIPS Actually Did)

- STRIPS used GPS as a search algorithm:
- Newell and Simon's `General Problem Solver`
  - Determine difference between situation and goal
  - Select an operator that reduces the difference
  - Apply the operator
  - Reconsider goal in new situation

STRIPS Algorithm (con’t)

- Try to prove the goal in a situation being considered.
  - STRIPS used resolution
- If fail, use incomplete proof as the difference.
- Use Add and Del lists of operators to pick on that may help the proof continue.
  - Heuristic: Look for operator that may help resolve something.
  - Turn the preconditions of the operators into new subgoals.
  - Recursively attempt to achieve them.
- Apply operator by modifying the KB.
  - Can reversibly modify the KB, or use Add list as additions and Del list as filters without changing KB.

STRIPS and the Frame Problem

- The computational part of the frame problem is now addressed. How?
  - Everything in initial situation is assumed to remain true unless it is mentioned on an Add or Del list.
  - Add, Del lists are relatively small compared to the KB, reflecting the fact that operators don’t change much in the world.
  - I.e., moving from one situation to another is essentially linear in number of operators.

Very Simple STRIPS Example

- Initial Situation:  \textbf{At(Robot,A)}
- Goal:  \textbf{At(Robot,B)}
- Operators:
  \begin{itemize}
  
  \item \textbf{Goto(m,n)}
    \begin{itemize}
      \item Add:  \textbf{At(Robot,n)}
      \item Del:  \textbf{At(Robot,m)}
      \item Pre:  \textbf{At(Robot,m)}
    \end{itemize}
  \item \textbf{Push(k,m,n)}
    \begin{itemize}
      \item Add:  \textbf{At(Robot,n)} \land \textbf{At(k,n)}
      \item Del:  \textbf{At(Robot,m)} \land \textbf{At(k,m)}
      \item Pre:  \textbf{At(Robot,m)} \land \textbf{At(k,m)} \land \text{Box(k)}
    \end{itemize}
  \end{itemize}
Simple STRIPS Example Goal: \(\text{At(\text{Robot},B)}\)

- Add negated goal \(\neg\text{At(\text{Robot},B)}\) to KB.
- Proof attempt fails, leaving \(\neg\text{At(\text{Robot},B)}\) to resolve away.
- Look for operator to reduce difference, i.e., complete proof.
  - \(\text{Goto}(m,n)\) looks like it will help the proof.
    - because it adds \(\text{At(\text{Robot},n)}\), which unifies with negation of \(\neg\text{At(\text{Robot},B)}\), suggesting the (partially instantiated) operator \(\text{Goto}(m,B)\).
    - Actually, so does \(\text{Push}(k,m,n)\); heuristic helps select former.
- Precondition becomes new subgoal: \(\text{At(\text{Robot},m)}\)
- Can prove this, because \(\text{At(\text{Robot},A)}\)
- Can now execute (fully instantiated operator) \(\text{Goto}(A,B)\).
- Simulate execution: Hack KB by adding Adds (\(\text{At(\text{Robot},B)}\)) and deleting Dels (\(\text{At(\text{Robot},A)}\)).
- Afterwards, KB contains \(\text{At(\text{Robot},B)}\) (and not \(\text{At(\text{Robot},A)}\)).
- Now can complete proof, so done!
- Plan is \(\text{Goto}(A,B)\).

STRIPS Example (con’t)

- Negated goal in clausal form:
  \[\neg\text{At(\text{Box1},x)},\neg\text{At(\text{Box2},x)},\neg\text{At(\text{Box3},x)}\]
- Resolve with any of the box location predications:
  \[\neg\text{At(\text{Box1},x)},\neg\text{At(\text{Box2},x)},\neg\text{At(\text{Box3},x)}\]
  \[\text{At(\text{Box1},B)}\]
  \[\neg\text{At(\text{Box2},B)},\neg\text{At(\text{Box3},B)}\]
- This is our incomplete proof.
- Select operator that might let proof continue.
  - Add list of \(\text{Push}(k,m,n)\) contains \(\text{At}(k,n)\), which looks promising:
    \[\neg\text{At(\text{Box2},B)},\neg\text{At(\text{Box3},B)}\]
    \[\text{At}(k,n)\]
    \[\neg\text{At(\text{Box3},B)}\]
- With unifier \((k/\text{Box2},n/B)\)
  giving us (partially instantiated) operator \(\text{OP1}: \text{Push}(\text{Box2},m,B)\)
- Moreover, Del list is \(\text{At(\text{Box2},m)} \land \text{At(\text{Robot},m)})\); nothing resembling either conjunct is used in the proof so far.

More Interesting STRIPS Example

- Initial Situation \(\mathbf{S_0}\):
  \(\text{Box(\text{Box1})} \land \text{Box(\text{Box2})} \land \text{Box(\text{Box3})}\)
  \(\land \text{At(\text{Robot},A)}\)
  \(\land \text{At(\text{Box1},B)} \land \text{At(\text{Box2},C)} \land \text{At(\text{Box3},D)}\)
- Initial Goal \(\mathbf{G_0}\):
  \(\exists x \text{At(\text{Box1},x)} \land \text{At(\text{Box2},x)} \land \text{At(\text{Box3},x)}\)
  (I.e., all three boxes should be at the same place.)
- For this example, we'll use the notation \(\mathbf{S_i}, \mathbf{G_n}, \mathbf{G_{n-1}}\ldots\) to designate being in situation \(\mathbf{S_i}\) and have goal stack \(\mathbf{G_n}, \mathbf{G_{n-1}}\ldots\).
- So the initial state of affairs is \(\mathbf{S_0}, \mathbf{G_0}\).

STRIPS Example (con’t)

- New goal is precondition of \(\text{Push(\text{Box2},m,B)}\):
  \(\mathbf{G_1}: \text{At(\text{Box2},m)} \land \text{At(\text{Robot},m)} \land \text{Box(\text{Box2})}\)
  State of affairs is \(\mathbf{S_0}(\mathbf{G_0},\mathbf{G_1})\)
- Negated goal is
  \[\neg\text{At(\text{Box2},m)},\neg\text{At(\text{Robot},m)},\neg\text{Box(\text{Box2})}\]
  - In KB is \(\text{Box(\text{Box2})}\), so can cancel out \(\neg\text{Box(\text{Box2})}\).
  - Since \(\text{At(\text{Robot},A)}\), can deduce \(\neg\text{At(\text{Box2},A)}\).
  - Since \(\text{At(\text{Box2},C)}\), can deduce \(\neg\text{At(\text{Robot},C)}\).
- Stuck; need strategy for operator selection.
  - First suggests pushing \(\text{Box2}\) again, a bad idea.
  - Second suggests \(\text{Goto}\) operator; less objectionable.
  - Specifically:
    - Operator \(\text{OP2}: \text{Goto}(\text{m,C})\)
    - Precondition \(\text{At(\text{Robot},m)}\) becomes new goal \(\mathbf{G_2}\).
    - State of affairs is \(\mathbf{S_0}(\mathbf{G_2},\mathbf{G_1},\mathbf{G_0})\)
• But we can prove $G_2$: $\text{At(Robot,m)}$, because $\text{At(Robot,A)}$ is true.
• Apply (fully instantiated) $\text{OP}_2$: $\text{Goto(A,C)}$ to produce new situation, $S_1$

$\text{Now we can prove } G_1, \text{ and hence apply } \text{OP}_1 \text{ getting situation } S_2.$
But we can prove $G_2$: $\text{At(Robot,m)}$, because $\text{At(Robot,A)}$ is true.

- Apply (fully instantiated) $\text{OP}_2$: $\text{Goto(A,C)}$ to produce new situation, $S_1$
  - Add $\text{At(Robot,C)}$
  - Delete $\text{At(Robot,A)}$
  - State of affairs is $(S_1, (G_1,G_0))$

$\textbf{Now we can prove } G_1$, and hence apply $\text{OP}_1$ getting situation $S_2$
  - State of affairs is $(S_2, (G_0))$

$\textbf{And so on.}$

**STRIPS Example (con't)**

**STRIPS Running Time**

- STRIPS spent most of its time in the theorem prover:

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>TP time</th>
<th>No. of nodes in sol.</th>
<th>No. of operators in sol.</th>
<th>No. of nodes in search tree</th>
<th>No. of operators in search tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 boxes</td>
<td>66</td>
<td>49.6</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>M&amp;B</td>
<td>113</td>
<td>83</td>
<td>13</td>
<td>21</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

- It didn't necessarily find a good solution.
  - For M&B problem: get on box, get off box, push box under bananas, get on box, get bananas

- It didn't guarantee finding a solution.

- It was inefficient for handling conjunctive subgoals.
  - We'll return to this problem later.

**Summary**

- It is more efficient to represent actions by how they modify a KB, than by using the purely logical situation calculus.
- We can do so using a STRIPS-style operator representation.
- This uses add and delete lists.
- STRIPS avoids the need for frame axioms.