PLANNING
1 What is Planning?

- Key problem facing agent is deciding what to do.
- We want agents to be taskable: give them goals to achieve, have them decide for themselves how to achieve them.
- Basic idea is to give an agent:
  - representation of goal to achieve;
  - knowledge about what actions it can perform; and
  - knowledge about state of the world;
and to have it generate a plan to achieve the goal.
- Essentially, this is automatic programming.
• Question: How do we represent…
  – goal to be achieved;
  – state of environment;
  – actions available to agent;
  – plan itself.

• All this can be done in first-order logic…
• We’ll illustrate the techniques with reference to the *blocks world*.
• Contains a robot arm, 3 blocks (A, B and C) of equal size, and a table-top.
• Initial state:
To represent this environment, need an ontology.

- $On(x, y)$: obj $x$ on top of obj $y$
- $OnTable(x)$: obj $x$ is on the table
- $Clear(x)$: nothing is on top of obj $x$
- $Holding(x)$: arm is holding $x$
Here is a FOL representation of the blocks world described above:

\[
\begin{align*}
\text{Clear}(A) \\
\text{On}(A, B) \\
\text{OnTable}(B) \\
\text{OnTable}(C) \\
\text{Clear}(C)
\end{align*}
\]

Use the \textit{closed world assumption}: anything not stated is assumed to be \textit{false}.
• A goal is represented as a FOL formula.
• Here is a goal:

\[ \text{OnTable}(A) \land \text{OnTable}(B) \land \text{OnTable}(C) \]

• Which corresponds to the state:

\[
\begin{array}{c}
\text{A} \\
\text{B} \\
\text{C}
\end{array}
\]

• Actions are represented using a technique that was developed in the STRIPS planner.
Each action has:

– a name
  which may have arguments;
– a pre-condition list
  list of facts which must be true for action to be executed;
– a delete list
  list of facts that are no longer true after action is performed;
– an add list
  list of facts made true by executing the action.

Each of these may contain variables.
Example 1:
The *stack* action occurs when the robot arm places the object $x$ it is holding is placed on top of object $y$.

$$Stack(x, y)$$
pre $Clear(y) \land Holding(x)$
del $Clear(y) \land Holding(x)$
add $ArmEmpty \land On(x, y)$
Example 2:
The *unstack* action occurs when the robot arm picks an object \( x \) up from on top of another object \( y \).

\[
\text{UnStack}(x, y) \\
\text{pre} \quad \text{On}(x, y) \land \text{Clear}(x) \land \text{ArmEmpty} \\
\text{del} \quad \text{On}(x, y) \land \text{ArmEmpty} \\
\text{add} \quad \text{Holding}(x) \land \text{Clear}(y)
\]

Stack and UnStack are *inverses* of one-another.
• Example 3:
The *pickup* action occurs when the arm picks up an object \( x \) from the table.

\[
\begin{align*}
\text{Pickup}(x) \\
\text{pre } & \text{Clear}(x) \land \text{OnTable}(x) \land \text{ArmEmpty} \\
\text{del } & \text{OnTable}(x) \land \text{ArmEmpty} \\
\text{add } & \text{Holding}(x)
\end{align*}
\]

• Example 4:
The *putdown* action occurs when the arm places the object \( x \) onto the table.

\[
\begin{align*}
\text{PutDown}(x) \\
\text{pre } & \text{Holding}(x) \\
\text{del } & \text{Holding}(x) \\
\text{add } & \text{Holding}(x) \land \text{ArmEmpty}
\end{align*}
\]
• What is a plan?
  A sequence (list) of actions, with variables replaced by constants.

• So, to get from:

\[
\begin{array}{c}
A \\
\hline
B \\
\hline
C \\
\end{array}
\quad \text{to} \quad
\begin{array}{c}
\hline
A \\
B \\
\hline
C \\
\end{array}
\]
• We need the set of actions:

\begin{align*}
Unstack(A) \\
Putdown(A) \\
Pickup(B) \\
Stack(B, C) \\
Pickup(A) \\
Stack(A, B)
\end{align*}
• In “real life”, plans contain conditionals (IF .. THEN...) and loops (WHILE... DO...), but most simple planners cannot handle such constructs — they construct linear plans.

• Simplest approach to planning: means-ends analysis.

• Involves backward chaining from goal to original state.

• Start by finding an action that has goal as post-condition. Assume this is the last action in plan.

• Then figure out what the previous state would have been. Try to find action that has this state as post-condition.

• Recurse until we end up (hopefully!) in original state.
function $plan($
  \textit{d} : \text{WorldDesc}, \quad \text{\textit{d} : WorldDesc, \quad // initial env state}$
  \textit{g} : \text{Goal}, \quad \text{\textit{g} : Goal, \quad // goal to be achieved}$
  \textit{p} : \text{Plan}, \quad \text{\textit{p} : Plan, \quad // plan so far}$
  \textit{A} : \text{set of actions} \quad \text{\textit{A} : set of actions \quad // actions available})$

1. if $\textit{d} \models \textit{g}$ then
2. \quad \text{return } \textit{p}$
3. else
4. \quad \text{choose } \textit{a} \text{ in } \textit{A} \text{ such that}$
5. \quad \quad \textit{add}(\textit{a}) \models \textit{g} \text{ and}$
6. \quad \quad \textit{del}(\textit{a}) \not\models \textit{g}$
7. \quad \quad \text{set } \textit{g} = \textit{pre}(\textit{a})$
8. \quad \quad \text{append } \textit{a} \text{ to } \textit{p}$
9. \quad \text{return } \textit{plan}(\textit{d}, \textit{g}, \textit{p}, \textit{A})$
• How does this work on the previous example?
This algorithm not guaranteed to find the plan…

… but it is sound: If it finds the plan is correct.

Some problems:

– negative goals;
– maintenance goals;
– conditionals & loops;
– exponential search space;
– logical consequence tests;
Sussman’s Anomaly

- Consider we have the following initial state and goal state:

```
  B  C
  A  B
  C  C
```

- What operations will be in the plan?
• Clearly we need to \textit{Stack} B on C at some point, and we also need to \textit{Unstack} A from C and \textit{Stack} it on B.

• Which operation goes first?

• Obviously we need to do the \textit{UnStack} first, and the \textit{Stack} B on C, but the planner has no way of knowing this.

• It also has no way of “undoing” a partial plan if it leads into a dead end.

• So if it chooses to \textit{Stack}(A, C) after the \textit{Unstack}, it is sunk.

• This is a big problem with linear planners

• How could we modify our planning algorithm?
• Modify the middle of the algorithm to be:

1. if $d \models g$ then
2. return $p$
3. else
4. choose $a$ in $A$ such that
5. \[ add(a) \models g \text{ and} \]
6. \[ del(a) \not\models g \]
6a. \[ no\_clobber(add(a), del(a), rest\_of\_plan) \]
7. set $g = pre(a)$
8. append $a$ to $p$
9. return $plan(d, g, p, A)$
Summary

- This lecture briefly introduced simple (classical planning).
- It showed how to use means-ends analysis to create a linear plan when the world is represented using STRIPS operators.
- We also talked about some of the problems with this approach.
- In particular we talked about Sussman’s anomaly, which leads us to partial order planning, the topic of the next lecture.