Planning Part 2

Lecture 6
November 4, 2008
News

• Revised proposals due tonight (before morning)

• No readings this week (work on projects)

• Office hours tomorrow also Monday (but not next wed)
The monkey-and-bananas problem is faced by a monkey in a laboratory with the bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at A, the bananas at B, and the box at C. The monkey and box have height Low, but if the monkey climbs onto the box he will have height High, the same as the bananas. The actions available to the monkey include Go from one place to another, Push an object from one place to another, ClimbUp onto or ClimbDown from an object, and Grasp or Ungrasp an object. Grasping results in holding the object if the monkey and object are in the same place at the same height.

Write down the initial state description

Write down STRIPS-style definitions of the six actions

Suppose the monkey wants to fool the scientists by grabbing the bananas but leaving the box in its original place. Write this as a general goal (not assuming the box is at C). Can this goal be solved by a STRIPS-style system?
Answer

a. The initial state is:

\[
\text{At(Monkey, A)} \land \text{At(Bananas, B)} \land \text{At(Box, C)} \land \\
\text{Height(Monkey, Low)} \land \text{Height(Box, Low)} \land \text{Height(Bananas, High)} \land \\
\text{Pushable(Box)} \land \text{Climbable(Box)}
\]

b. The actions are:

\[
\text{Action}\text{(ACTION}: \text{Go}(x, y), \text{PRECOND}: \text{At(Monkey, x)}, \\
\text{EFFECT}: \text{At(Monkey, y)} \land \neg(\text{At(Monkey, x)}))
\]

\[
\text{Action}\text{(ACTION}: \text{Push}(b, x, y), \text{PRECOND}: \text{At(Monkey, x)} \land \text{Pushable(b)}, \\
\text{EFFECT}: \text{At(b, y)} \land \text{At(Monkey, y)} \land \neg\text{At(b, x)} \land \neg\text{At(Monkey, x)}))
\]

\[
\text{Action}\text{(ACTION}: \text{ClimbUp}(b), \\
\text{PRECOND}: \text{At(Monkey, x)} \land \text{At(b, x)} \land \text{Climbable(b)}, \\
\text{EFFECT}: \text{On(Monkey, b)} \land \neg\text{Height(Monkey, High)})
\]

\[
\text{Action}\text{(ACTION}: \text{Grasp}(b), \\
\text{PRECOND}: \text{Height(Monkey, h)} \land \text{Height(b, h)} \\
\land \text{At(Monkey, x)} \land \text{At(b, x)}, \\
\text{EFFECT}: \text{Have(Monkey, b)})
\]

\[
\text{Action}\text{(ACTION}: \text{ClimbDown}(b), \\
\text{PRECOND}: \text{On(Monkey, b)} \land \text{Height(Monkey, High)}, \\
\text{EFFECT}: \neg\text{On(Monkey, b)} \land \neg\text{Height(Monkey, High)} \\
\land \text{Height(Monkey, Low)}
\]

\[
\text{Action}\text{(ACTION}: \text{UnGrasp}(b), \text{PRECOND}: \text{Have(Monkey, b)}, \\
\text{EFFECT}: \neg\text{Have(Monkey, b)})
\]

c. In situation calculus, the goal is a state \( s \) such that:

\[
\text{Have(Monkey, Bananas, s)} \land (\exists x \text{ At(Box, x, s_0)} \land \text{At(Box, x, s)})
\]

In STRIPS, we can only talk about the goal state; there is no way of representing the fact that there must be some relation (such as equality of location of an object) between two states within the plan. So there is no way to represent this goal.
Partially ordered plans

A plan is **complete** iff every precondition is achieved.

A precondition is **achieved** iff it is the effect of an earlier step and no possibly intervening step undoes it.
Solutions

• Consistent plan - plan with no cycles in ordering constraints and no conflicts with causal links

• Solution - Consistent plan with no open preconditions
POP as Search

- Initial plan contains *Start* and *Finish*, all preconditions in *Finish* are open preconditions

- Successor function picks open precondition p on an action B and generates a successor plan for every consistent way to choose action A that achieves p

- Casual link a -> B, p and ordering constraint A<B are added to plan

- Resolve conflicts between new link and existing actions and new action A and all causal links. Conflict between A->B,p and C is resolved by adding B<C or C<A. Add successor states for either or both if consistent.

- Goal test checks if solution (no open preconditions)
How to solve?

A plan is **complete** iff every precondition is achieved.

A precondition is **achieved** iff it is the effect of an earlier step and no **possibly intervening** step undoes it.
Heuristics for POP?
Heuristics

- PRO: easy to decompose into subproblems
- CON: no direct representation of states, no notion of how close to goal
- Heuristic: number of open preconditions
- Most-constrained variable heuristic - pick open condition that can be satisfied in the fewest number of ways
Planning Graphs

- Sequence of levels where level 0 is initial state
- Each level contains literals and actions
  - literals are those that could be true at time step (level) depending on actions at previous levels
- Also consider inactions
- Record mutex links between literals that are mutually exclusive
- Requires propositional planning problems (no variables)
- When state $S_i$ and Action $A_i$ are identical, the graph has leveled off
Have cake and eat cake too problem

Init(Have(Cake))
Goal(Have(Cake) ^ Eaten(Cake))
Action(Eat(Cake))
  PRECOND: Have(Cake)
  EFFECT: ~Have(Cake) ^ Eaten(Cake)
Action(Bake(Cake))
  PRECOND: ~Have(Cake)
  EFFECT: Have(Cake)
Air transport problem

Init(At(C_1, SFO) ^ At(P_1, SFO) ^ At(P_2, JFK) ^ Cargo(C_1) ^ Cargo(C_1) ^ Cargo(C_2) ^ Plane(P_1) ^ Plane(P_2) ^ Airport(JFK) ^ Airport(SFO))

Goal(At(C_1,JFK) ^ At(C_2,SFO))

Action(Load(c,p,a),
   PRECOND: At(c,a) ^ At(p,a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)
   EFFECT: ~At(c,a) ^ In(c,p))

Action(Unload(c,p,a),
   PRECOND: In(c,p) ^ At(p,a) ^ Cargo(c) ^ Plane(p) ^ Airport(a)
   EFFECT: At(c,a) ^ ~In(c,p))

Action(Fly(p,from,to),
   PRECOND: At(p, from) ^ Plane(p) ^ Airport(from) ^ Airport(to)
   EFFECT: ~At(p,from) ^ At(p,to))

Construct levels 0, 1, 2 for this problem
function GRAPHPLAN(problem) returns solution or failure

graph = INITIAL-PLANNING-GRAPH(problem)
goals = GOALS[problem]

loop do
    if goals all non-mutex in last level of graph then do
        solution = EXTRACT-SOLUTION(graph, goals, LENGTH(graph))
        if solution != failure then return solution
        else if NO-SOLUTION-POSSIBLE(graph) then return failure
    end if
end if

graph = EXPAND-GRAPH(graph, problem)
SATPlan

- Translate a planning problem into propositional axioms
- Apply a satisfiability algorithm to find a model that corresponds to a valid plan
- Choosing the right representation affects the plan
- Goal state must be associated with a particular time
Representation Issues

• Have to specify false states (open world)
• Need to add precondition axioms (conditionals) to prevent illegal actions
• Need to represent mutex links in planning graph
  • Action exclusion axioms
    • (and statements \( x \land \neg y, y \land \neg x \))
  • State constraints (more general statements, hard to write)
Distributed Planning and Problem Solving

- Examples of:
  - Centralized planning for centralized action
  - Centralized planning for distributed action
  - Distributed planning for centralized action
  - Distributed planning for distributed action
Challenges of distributed vs centralized planning
Security issues in Distributed planning?

• What about in the joint intention framework?
On Acting Together

• How useful is the joint commitment framework?

• How might you extend it?
Create an expression using the logic of on acting together

- Then pass your expression to the other groups to be interpreted
Reviewing “On Acting Together”

• How would this paper be evaluated by our course criteria?

• What should the standards be for this sort of paper?

• Hindsight: 375 citations (google scholar)