Planning (Chapter 10)

Planning

• Problem: I’m at home and I need milk, bananas, and a drill.

• How is planning different from regular search?
  – States and action sequences typically have complex internal structure
  – State space and branching factor are huge
  – Multiple subgoals at multiple levels of resolution

• Examples of planning applications
  – Scheduling of tasks in space missions
  – Logistics planning for the army
  – Assembly lines, industrial processes
  – Robotics
  – Games, storytelling
A representation for planning

- **STRIPS** (Stanford Research Institute Problem Solver): classical planning framework from the 1970s
- **States** are specified as conjunctions of predicates
  - Start state: \( \text{At(home)} \land \text{Sells(SM, Milk)} \land \text{Sells(SM, Bananas)} \land \text{Sells(HW, drill)} \)
  - Goal state: \( \text{At(home)} \land \text{Have(Milk)} \land \text{Have(Banana)} \land \text{Have(drill)} \)
- **Actions** are described in terms of preconditions and effects:
  - \( \text{Go(x, y)} \)
    - **Precond:** \( \text{At(x)} \)
    - **Effect:** \( \neg \text{At(x)} \land \text{At(y)} \)
  - \( \text{Buy(x, store)} \)
    - **Precond:** \( \text{At(store)} \land \text{Sells(store, x)} \)
    - **Effect:** \( \text{Have(x)} \)
- Planning is “just” a search problem
Challenges of planning: “Sussman anomaly”

Start state:

Goal state:

Let’s try to achieve On(A,B):

Let’s try to achieve On(B,C):

http://en.wikipedia.org/wiki/Sussman_Anomaly
Challenges of planning: “Sussman anomaly”

• Shows the limitations of non-interleaved planners that consider subgoals in sequence and try to satisfy them one at a time
  – If you try to satisfy subgoal X and then subgoal Y, X might undo some preconditions for Y, or Y might undo some effects of X

• More powerful planning approaches must interleave the steps towards achieving multiple subgoals

http://en.wikipedia.org/wiki/Sussman_Anomaly
Algorithms for planning

- **Forward (progression) state-space search**: starting with the start state, find all applicable actions (actions for which preconditions are satisfied), compute the successor state based on the effects, keep searching until goals are met
  - Can work well with good heuristics
Algorithms for planning

- **Forward (progression) state-space search**: starting with the start state, find all applicable actions (actions for which preconditions are satisfied), compute the successor state based on the effects, keep searching until goals are met
  - Can work well with good heuristics
- **Backward (regression) relevant-states search**: to achieve a goal, what must have been true in the previous state?
Situation space planning vs. plan space planning

- **Situation space planners:** each node in the search space represents a world state, arcs are actions in the world
  - Plans are sequences of actions from start to finish
  - Must be *totally ordered*

- **Plan space planners:** nodes are (incomplete) plans, arcs are transformations between plans
  - Actions in the plan may be *partially ordered*
  - **Principle of least commitment:** avoid ordering plan steps unless absolutely necessary
Partial order planning

- Task: put on socks and shoes
Partial Order Planning Example

Start: empty plan

Action: find flaw in the plan and modify plan to fix the flaw
Partial Order Planning Example

Start

Sells(SM, Milk)  At(Home)  Sells(SM, Bananas)

Sells(x1, Milk)  At(x1)  Sells(x2, Bananas)

Buy(x1, Milk)  Have(Milk)  Buy(x2, Bananas)  Have(Bananas)

At(x2)  Sells(x2, Bananas)  At(x3)  Go(x3, SM)  At(SM)  At(x3)

At(Home)  Have(Milk)  Have(Bananas)

x1 = SM  x2 = SM  x3 = Home

Finish
Application of planning: Automated storytelling

https://research.cc.gatech.edu/inc/mark-riedl
Application of planning: Automated storytelling

• Applications
  – Personalized experience in games
  – Automatically generating training scenarios (e.g., for the army)
  – Therapy for kids with autism
  – Computational study of creativity

https://research.cc.gatech.edu/inc/mark-riedl
Complexity of planning

• Planning is *PSPACE-complete*
  – The length of a plan can be exponential in the number of “objects” in the problem!

• Example: towers of Hanoi
Complexity of planning

• Planning is \textit{PSPACE}-complete
  – The length of a plan can be exponential in the number of “objects” in the problem!
  – So is game search

• Archetypal PSPACE-complete problem: \textit{quantified boolean formula} (QBF)
  – Example: is this formula true?
    \[
    \exists x_1 \forall x_2 \exists x_3 \forall x_4 (x_1 \lor \neg x_3 \lor x_4) \land (\neg x_2 \lor x_3 \lor \neg x_4)
    \]

• Compare to SAT:
  \[
  \exists x_1 \exists x_2 \exists x_3 \exists x_4 (x_1 \lor \neg x_3 \lor x_4) \land (\neg x_2 \lor x_3 \lor \neg x_4)
  \]

• Relationship between SAT and QBF is akin to the relationship between puzzles and games
Real-world planning

• Resource constraints
• Actions at different levels of granularity: hierarchical planning
• Incorporating sensing and feedback
• Contingencies: actions failing
• Uncertainty