Planning II

CSE 473

Logistics

- Tournament!
- PS3 - later today

Non programming exercises
Programming component: (mini project)
  - SPAM detection
  - Bag of words representation
  - Decision tree induction
  - Ensembles
  - Naive Bayes
  - Experiments & writeup

- Due in stages thru quarter's end
- New teams of 2 - due today

Generative Planning

Input
- Description of (initial state of) world (in some KR)
- Description of goal (in some KR)
- Description of available actions (in some KR)

Output
- Controller
  - E.g. Sequence of actions
  - E.g. Plan with loops and conditionals
  - E.g. Policy = f: states -> actions

Simplifying Assumptions

- Environment
  - Static vs. Dynamic
  - Fully Observable vs. Partially Observable
  - Deterministic vs. Stochastic

- Percepts vs. Actions
  - What action next?

STRIPS Action Schemata

- Instead of defining ground actions: pickup-A and pickup-B and ...
- Define a schema:

```
(:operator pickup
  :parameters ((block ?ob1))
  :precondition (and (clear ?ob1)
                   (on-table ?ob1)
                   (arm-empty))
  :effect (and (not (clear ?ob1))
            (not (on-table ?ob1))
            (not (arm-empty))
            (holding ?ob1)))
```

Outline

- Defn Planning
- STRIPS representation & assumptions
- World-space search
  - Forward chaining
  - Backwards chaining
  - Heuristics
- Graphplan
- SATplan
Backward-Chaining Search Thru Space of Partial World-States

- Problem: Many possible goal states are equally acceptable.
- From which one does one search?

Initial State is completely defined

Regression

- Let \( G \) be a KR sentence (e.g. in logic)
- Regressing a goal, \( G \), thru an action, \( A \)
- Yields the weakest precondition \( G' \)
  Such that: if \( G' \) is true before \( A \) is executed \( G \) is guaranteed to be true afterwards

Regression Example

\[
G' \rightarrow A \rightarrow G
\]

\[
\begin{align*}
& \neg \text{(clear } C) \land \text{(on-table } C) \land \text{(arm-empty)} \\
\end{align*}
\]

\[
\begin{align*}
\text{pick-up : parameters ((block ?ob1))} \\
\text{precondition (and (clear } ?ob1) \land \text{(on-table } ?ob1) \land \text{(arm-empty))} \\
\text{effect (and (holding } ?ob1)) \\
\end{align*}
\]

Conditional Effects

\[
\begin{align*}
\text{move-briefcase (Loc New)} \\
\text{precondition (and (at briefcase Loc) (location Loc))} \\
\text{effect (and (at Loc) (not (at briefcase Loc)))} \\
\text{(when (in paycheck briefcase) (and (at paycheck bank)) (not (at keys bank))) (not (at keys bank)))} \\
\end{align*}
\]

Heuristics for Regression

- Suppose: searching backwards to init state
- Want heuristic
  Under-estimate of distance from Init state to set of states represented by \( G \)
Outline
• Defn Planning
• STRIPS representation & assumptions
• World-space search
• Graphplan
  - Planning Graph Expansion
    • Mutex relations
  - Solution Extraction
• SATplan

Graphplan
• Phase 1 - Graph Expansion
  Necessary (insufficient) conditions for plan existence
  Local consistency of plan-as-CSP
• Phase 2 - Solution Extraction
  Variables
    • action execution at a time point
  Constraints
    • goals, subgoals achieved
    • no side-effects between actions

The Plan Graph

Graph Expansion

Constructing the planning graph...
• Initial proposition layer
  Just the initial conditions
• Action layer i
  If all of an action's preconds are in i-1
  Then add action to layer i
  Nop actions have P as precond and effect
• Proposition layer i+1
  For each action at layer i
  Add all its effects at layer i

Mutual Exclusion
• Two actions are mutex if
  • one clobbers the other's effects or preconditions
  • they have mutex preconditions
• Two proposition are mutex if
  • one is the negation of the other
  • all ways of achieving them are mutex
**Mutual Exclusion**

- *Actions* $A,B$ **exclusive (at a level)** if
  - $A$ deletes $B$'s precond, or
  - $B$ deletes $A$'s precond, or
  - $A$ & $B$ have inconsistent preconds

- *Propositions* $P,Q$ **inconsistent (at a level)** if
  - all ways to achieve $P$
  - exclude all ways to achieve $Q$

**Graphplan**

- Create level 0 in planning graph
- **Loop**
  - If goal $\subseteq$ contents of highest level (nonmutex)
    - Then search graph for solution
  - Else extend graph one more level

A kind of double search: forward direction checks
necessary (but insufficient) conditions for a solution,
Backward search verifies...

**Searching for a Solution**

- For each goal $G$ at time $t$
  - For each action $A$ making $G$ true at $t$
    - Select $A$ unless mutex with previously chosen action
    - If no actions work, backup to last $G$ (breadth first search)
  - Recurse on
    - preconditions of actions selected, $t-1$

**Planning Graph**

- Dinner Date
  - Initial Conditions: $\langle \text{and} (\text{cleanHands}) (\text{quiet}) \rangle$
  - Goal: $\langle \text{and} (\neg \text{noGarbage}) (\text{dinner}) (\text{present}) \rangle$
  - Actions:
    - (operator $\text{carry}$ :precondition $\langle \text{effect} (\text{and} (\neg \text{noGarbage}) (\neg \text{cleanHands})) \rangle$
    - (operator $\text{dolly}$ :precondition $\langle \text{effect} (\text{and} (\neg \text{noGarbage}) (\neg \text{quiet})) \rangle$
    - (operator $\text{cook}$ :precondition $\langle \text{effect} (\text{dinner}) \rangle$
    - (operator $\text{wrap}$ :precondition $\langle \text{effect} (\text{present}) \rangle$

- Planning Graph

  - noGarb
cleanH
dolly
cleanH
quiet
quiet
cook
quiet
wrap
quiet

  - dinner
  - present

  - 0 Prop 1 Action 2 Prop 3 Action 4 Prop
Are there any exclusions?

Do we have a solution?

Extend the Planning Graph

Searching Backwards

One (of 4) Possibilities

One (of 4) possibilities
Graphplan Solution Extraction as a Constraint Network

Not dolly & wrap
- Present
- Dinner

Not carry & cook
- NoGarb