CSP Planning; Logic Intro

CPSC 322 Lecture 17

February 20, 2006
Textbook §11.2 and §4.0 – 4.2

Lecture Overview

Recap

CSP Planning

Logic Intro

Forward Planning

Idea: search in the state-space graph.

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state s represent all of the actions that are legal in state s.
- A plan is a path from the state representing the initial state to a state that satisfies the goal.

Regression Planning

Idea: search backwards from the goal description: nodes correspond to subgoals, and arcs to actions.

- Nodes are propositions: partial assignments to state variables
- Start node: the goal condition
- Arcs correspond to actions
- A node that neighbours N via arc A is a variable assignment that specifies what must be true immediately before A so that N is true immediately after.
- The goal test is true if N is a proposition that is true of the initial state.

Regression Planning: defining nodes and arcs

- A node N is a partial assignment of values to variables:

$$[X_1 = v_1, \ldots, X_n = v_n]$$

- An action which can be taken to this node is one that achieves one of the $X_i = v_i$, and does not achieve any $X_j = v_j$ where v'_j is different from v_j.
- Any node that neighbours N via arc A must contain:
  - The prerequisites of action A
  - All of the elements of N that were not achieved by A

  N must be consistent.

Lecture Overview

Recap

CSP Planning

Logic Intro

Planning as a CSP

- We can go forwards and backwards at the same time, if we set up a planning problem as a CSP
- To do this, we need to “unroll” the plan for a fixed number of steps
  - this is called the horizon
- To do this with a horizon of k:
  - construct a variable for each feature at each time step from 0 to k
  - construct a boolean variable for each action at each time step from 0 to $k-1$.

CSP Planning: Constraints

As usual, we have to express the preconditions and effects of actions:

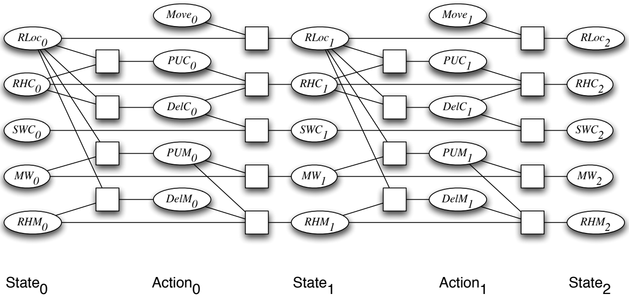
- precondition constraints
  - hold between state variables at time t and action variables at time t
  - specify when actions may be taken
- effect constraints
  - between state variables at time t, action variables at time t and state variables at time $t+1$
  - explain how state variables at time $t+1$ are affected by the action taken at time t
  - this includes both causal and frame axioms
    - basically, it goes back to the feature-centric representation we had before STRIPS
    - of course, solving the problem this way doesn't mean we can't *encode* the problem using STRIPS

CSP Planning: Constraints

Other constraints we must/may have:

- initial state constraints constrain the state variables at time 0
- goal constraints constrain the state variables at time k
- action constraints
  - specify which actions cannot occur simultaneously
    - note that without these constraints, there's nothing to stop the planner from deciding to take several actions simultaneously
    - when the order between several actions doesn't matter, this is a good thing
  - these are sometimes called mutual exclusion (mutex) constraints
- state constraints
  - hold between variables at the same time step
  - they can capture physical constraints of the system
  - they can encode maintenance goals

CSP Planning: Robot Example



Do you see why CSP planning is both forwards and backwards?

Lecture Overview

Recap

CSP Planning

Logic Intro

Logic: A more general framework for reasoning

- Let's now think about how to represent a world about which we have only partial (but certain) information
- Our tool: propositional logic
- General problem:
  - tell the computer how the world works
  - tell the computer some facts about the world
  - ask a yes/no question about whether other facts must be true

Why Propositions?

We'll be looking at problems that could still be represented using CSPs. Why use propositional logic?

- Specifying logical formulae is often more natural than filling in tables (i.e., arbitrary constraints)
- It is easier to check and debug formulae than tables
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries that may be more complicated than asking for the value of one variable
- It is easy to incrementally add formulae
- It can be extended to infinitely many variables (using logical quantification)
- This is a starting point for more complex logics (e.g., first-order logic) that go beyond CSPs.

Representation and Reasoning System

A Representation and Reasoning System (RRS) is made up of:

- syntax: specifies the symbols used, and how they can be combined to form legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: a (possibly nondeterministic) specification of how an answer can be produced.