Planning: Forward and CSP Planning

CPSC 322 – Planning 2

Textbook §11.2

Lecture Overview

1. Recap
2. Forward Planning
3. CSP Planning
4. Logic Intro

Planning

- With CSPs, we looked for solutions to essentially atemporal problems.
  - find a single variable assignment (state) that satisfies all of our constraints.

  Now consider a problem where we are given:
  - A description of an initial state
  - A description of the effects and preconditions of actions
  - A goal to achieve

  ...and want to find a sequence of actions that is possible and will result in a state satisfying the goal.
  - note: here we want not a single state that satisfies our constraints, but rather a sequence of states that gets us to a goal

Feature-Based Representation

- Features helped us to represent CSPs more compactly than states could.
  - The main idea: factor states into joint variable assignments
  - This allowed us to represent constraints compactly by discussing only relevant variables rather than full states.
- Now what we want to find is a sequence of variable assignments that goes from an initial state to a goal
- Instead of having one variable for every feature, we must instead have one variable for every feature at each time step, indicating the value taken by that feature at that time step!
- What do we need to build this representation?
  - the state space is easy: joint assignment to variables
  - initial state and goal state are also easy
  - the key is modeling actions

STRIPS Actions

- In STRIPS, an action has two parts:
  1. Precondition: a logical test about the features that must be true in order for the action to be legal
  2. Effects: a set of assignments to variables that are caused by the action

- If the feature V has value v after the action a has been performed, what can we conclude about a and/or the state of the world?
  - either $V = v$ was true in the state of the world immediately preceding execution of action a, or a sets $V = v$, or both.

Example

STRIPS representation of the action pick up coffee, PUC:

- **preconditions** $Loc = cs$ and $RHC = \overline{rhc}$
- **effects** $RHC = rhc$

STRIPS representation of the action deliver coffee, $DelC$:

- **preconditions** $Loc = off$ and $RHC = rhc$
- **effects** $RHC = \overline{rhc}$ and $SWC = \overline{swc}$

Note that Sam doesn't have to want coffee for Rob to deliver it; one way or another, Sam doesn't want coffee after delivery.

Lecture Overview

1 Recap

2 Forward Planning

3 CSP Planning

4 Logic Intro

Forward Planning

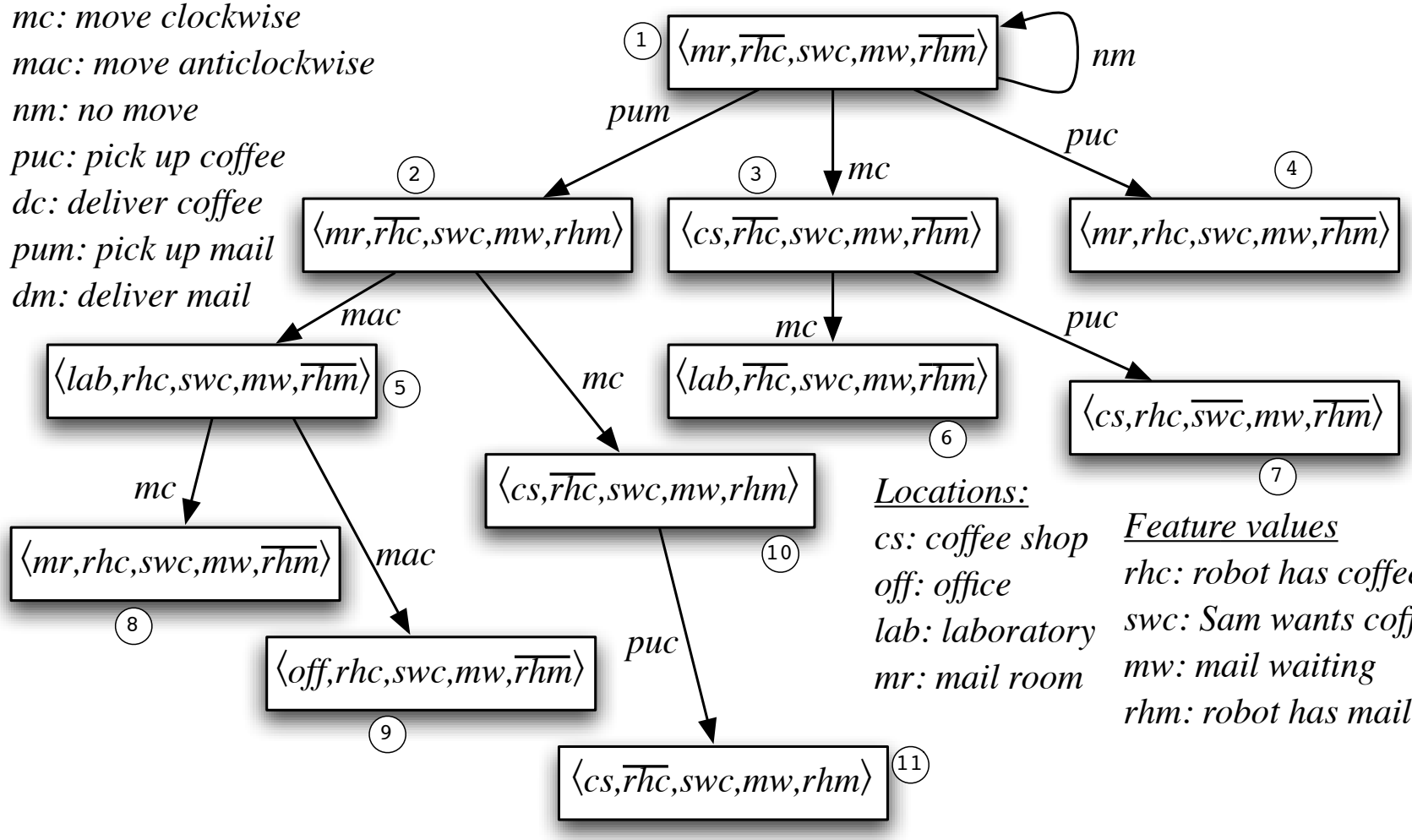
Idea: search in the state-space graph.

- 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.

What are the errors (none involve room locations)?

Actions

mc: move clockwise
mac: move anticlockwise
nm: no move
puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail



Locations:

cs: coffee shop
off: office
lab: laboratory
mr: mail room

Feature values

rhc: robot has coffee
swc: Sam wants coffee
mw: mail waiting
rhm: robot has mail

Improving Search Efficiency

Forward search can use domain-specific knowledge specified as:

- a heuristic function that estimates the number of steps to the goal
- domain-specific pruning of neighbors:
  - don't go to the coffee shop unless "Sam wants coffee" is part of the goal and Rob doesn't have coffee
  - don't pick-up coffee unless Sam wants coffee
  - unless the goal involves time constraints, don't do the "no move" action.

Lecture Overview

1 Recap

2 Forward Planning

3 CSP Planning

4 Logic Intro

Planning as a CSP

- We don't have to worry about searching forwards 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:

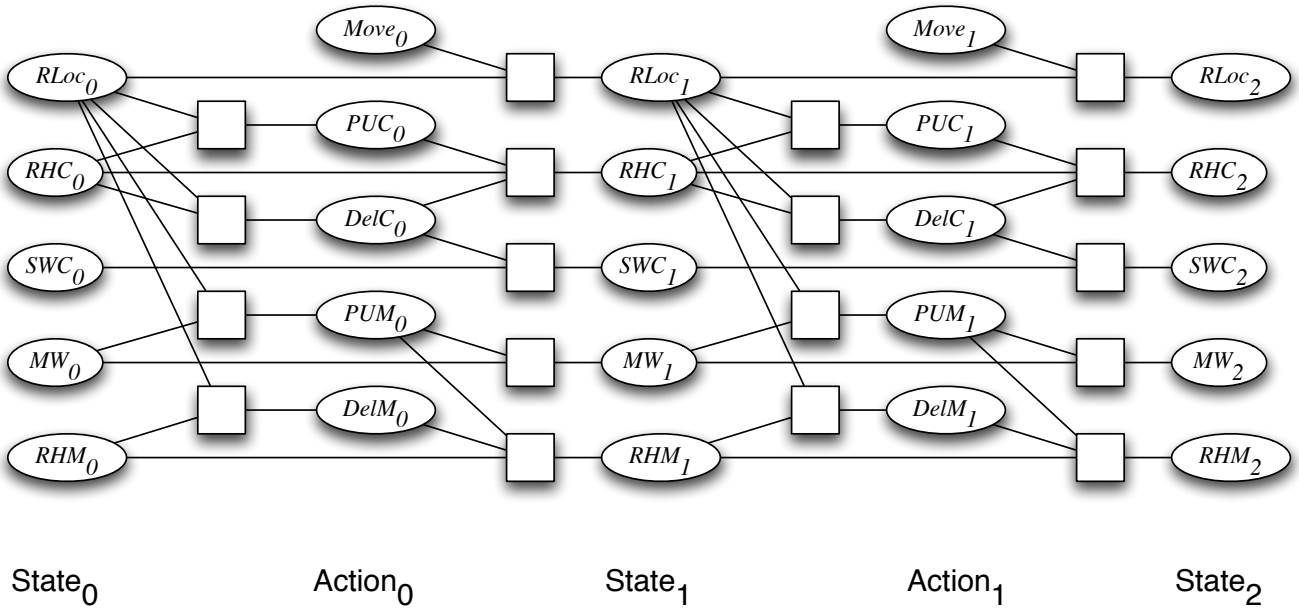
- precondition constraints
  - hold between state variables at time t and action variables at time t
  - specify when actions may be taken
- effect constraints
  - between state variables at time t, action variables at time t and state variables at time $t+1$
  - explain how state variables at time $t+1$ 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 the book discusses 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



The constraints shown represent the preconditions and the effects of actions.

Lecture Overview

1 Recap

2 Forward Planning

3 CSP Planning

4 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 constructing arbitrary constraints
- It is easier to check and debug formulae than constraints
- 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
- Logic 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 do go beyond CSPs.

Representation and Reasoning System

Definition (RSS)

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.

Using an RRS

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology).
3. Choose symbols in the computer to denote propositions
4. Tell the system knowledge about the domain.
5. Ask the system whether new statements about the domain are true or false.

Propositional Definite Clauses

- Propositional Definite Clauses: our first representation and reasoning system.
- Two kinds of statements:
  - that a proposition is true
  - that a proposition is true if one or more other propositions are true
- To define this RSS, we'll need to specify:
  - syntax
  - semantics
  - proof procedure