Chapter 6: Planning

Planning

- Planning is deciding what to do based on an agent's ability, its goals, and the state of the world.
- Initial assumptions:
  - The world is deterministic.
  - There are no exogenous events outside of the control of the robot that change the state of the world.
  - The agent knows what state it is in.
  - Time progresses discretely from one state to the next.
  - Goals are predicates of states that need to be achieved or maintained.
- Aim find a sequence of actions to solve a goal.

Classical Planning

- flat or modular or hierarchical
- explicit states or features or individuals and relations
- static or finite stage or indefinite stage or infinite stage
- fully observable or partially observable
- deterministic or stochastic dynamics
- goals or complex preferences
- single agent or multiple agents
- knowledge is given or knowledge is learned
- perfect rationality or bounded rationality

Outline

Representing Actions

- State-space Representation
- Feature-based Representation
- STRIPS Representation

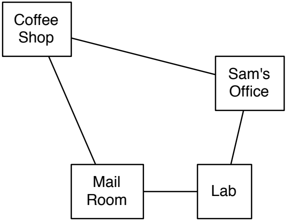
Planning

- Forward Planning
- Regression Planning
- Planning as a CSP

Actions

- A deterministic action is a partial function from states to states.
- The preconditions of an action specify when the action can be carried out.
- The effect of an action specifies the resulting state.

Delivery Robot Example



Features:

RLoc – Rob's location

RHC – Rob has coffee

SWC – Sam wants coffee

MW – Mail is waiting

RHM – Rob has mail

Actions:

mc – move clockwise

mcc – move counterclockwise

puc – pickup coffee

dc – deliver coffee

pum – pickup mail

dm – deliver mail

Explicit State-space Representation

State	Action	Resulting State
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	mc	$\langle mr, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	mcc	$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	dm	$\langle off, \overline{rhc}, swc, \overline{mw}, \overline{rhm} \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	mcc	$\langle cs, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	mc	$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
...	...	...

Feature-based representation of actions

For each action:

- **precondition** is a proposition that specifies when the action can be carried out.

For each feature:

- **causal rules** that specify when the feature gets a new value and
- **frame rules** that specify when the feature keeps its value.

Example feature-based representation

Precondition of "pick-up coffee" (puc):

$$RLoc{=}cs \land \overline{rhc}$$

Rules for "new location is coffee shop" (cs):

$$RLoc'{=}cs \leftarrow RLoc{=}off \land Act{=}mcc$$

$$RLoc'{=}cs \leftarrow RLoc{=}mr \land Act{=}mc$$

$$RLoc'{=}cs \leftarrow RLoc{=}cs \land Act \neq mcc \land Act \neq mc$$

Rules for "robot has coffee" (rhc)

$$rhc' \leftarrow rhc \land Act \neq dc$$

$$rhc' \leftarrow Act{=}puc$$

STRIPS Representation

Divide the features into:

- primitive features
- derived features. There are rules specifying how derived can be derived from primitive features.

For each action:

- precondition that specifies when the action can be carried out.
- effect a set of assignments of values to primitive features that are made true by this action.

STRIPS assumption: every primitive feature not mentioned in the effects is unaffected by the action.

Example STRIPS representation

Pick-up coffee (puc):

- precondition: $[cs, \overline{rhc}]$
- effect: $[rhc]$

Deliver coffee (dc):

- precondition: $[off, rhc]$
- effect: $[\overline{rhc}, \overline{swc}]$

Outline

- Representing Actions
  - State-space Representation
  - Feature-based Representation
  - STRIPS Representation
- Planning
  - Forward Planning
  - Regression Planning
  - Planning as a CSP

Planning

Given:

- A description of the effects and preconditions of the actions
- A description of the initial state
- A goal to achieve

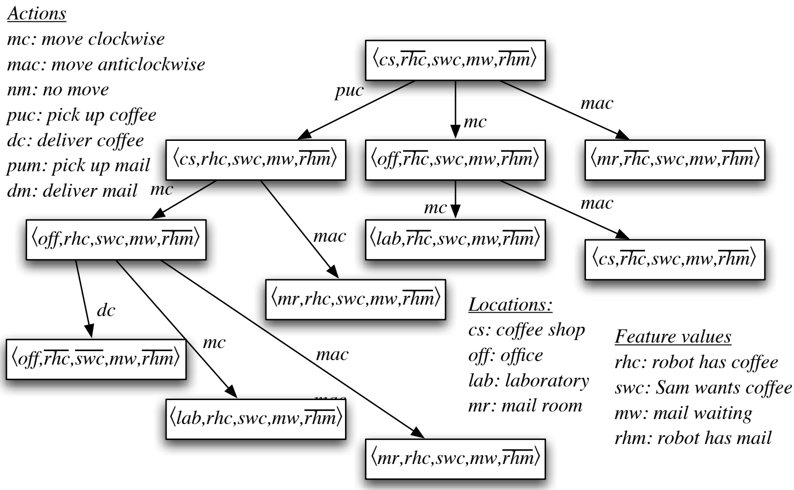
find a sequence of actions that is possible and will result in a state satisfying the goal.

Forward Planning

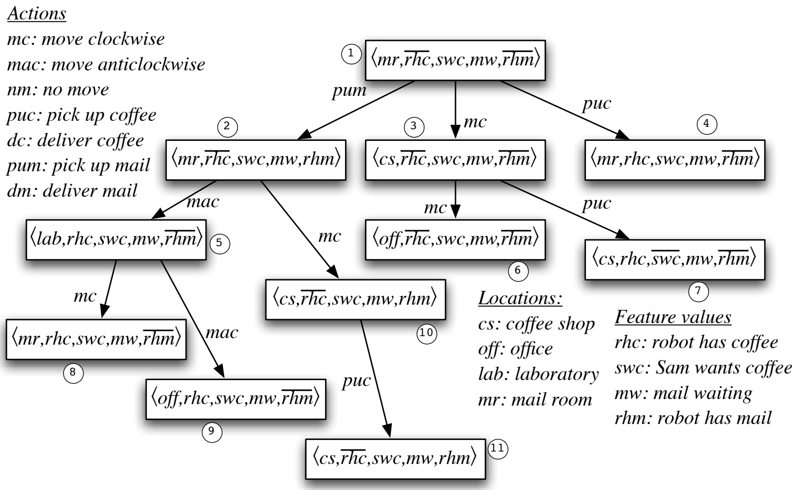
Idea: search in the state-space graph.

- 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.

Example state-space graph



What are the errors?



Forward planning representation

- The search graph can be constructed on demand: you only construct reachable states.
- If you want a cycle check or multiple-path pruning, you need to be able to find repeated states.
- There are a number of ways to represent states:
  - As a specification of the value of every feature
  - As a path from the start state

  Which one is better suited for multiple-path pruning?

Forward planning

- achievement goals: conditions on the final state of a plan
- maintenance goals: conditions that need to hold in any state of a plan, usually safety requirements:
  - keep away from the staircase,
  - always retain enough energy to be able to escape from unexpected difficulties, etc.
- transient goals: must be achieved somewhere in the plan but do not have to hold at the end
- resource goals: e.g. minimize energy consumption or time to reach the goal

Forward planning

A plan is a legal state sequence $s = s_1 s_2 ... s_n$ from the initial state s_1 to a goal state s_n, where

- the goal state s_n satisfies the achievement goals
- for all states $s_i \in s$ the preconditions of the action leading from state s_i to s_{i+1} are satisfied in state s_i
- no state $s_i \in s$ violates a maintenance goal
- the transient goals are satisfied in at least one state $s_i \in s$
- there is no other legal state sequence $s' = s_1 ... s_m$ to a (possibly different) goal state s_m that satisfies the resource goals in a better way

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.

Regression/Backward Planning

Idea: search backwards from the goal description until the initial state for the planning problem is reached

- Nodes correspond to (sub-)goals, they can represent several different states.
- Arcs correspond to actions.
- The initial state of the search is the goal state of the plan (and vice versa).

Regression/Backward Planning

- Nodes are propositions: a formula made up of assignments of values to features
- Arcs correspond to actions that can achieve one of the goals
- Neighbors of a node N associated with arc A specify what must be true immediately before A so that N is true immediately after.
- The start node of the search is the goal to be achieved by the plan.
- $goal(N)$ for the search procedure is true if N is a proposition that is true in the initial state of the planning problem.

Defining nodes and arcs

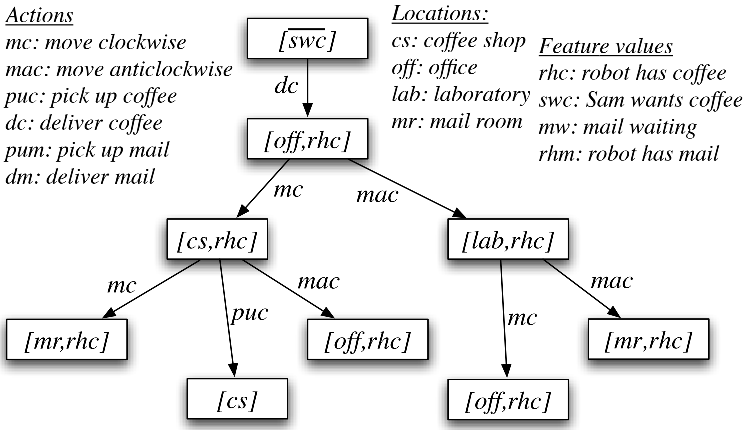
- A node N can be represented as a set of assignments of values to variables:

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

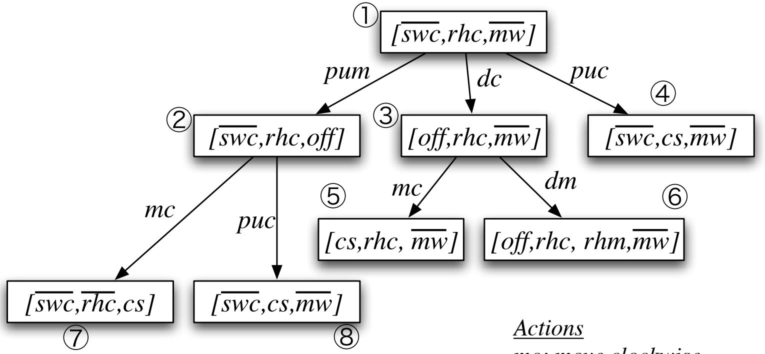
  This is a set of assignments you want to hold.
- The last action is one that achieves one of the $X_i = v_i$, and does not achieve $X_j = v_j'$ where v_j' is different to v_j.
- The neighbor of N along arc A must contain:
  - The prerequisites of action A
  - All of the elements of N that were not achieved by A

  N must be consistent.

Regression example



Find the errors



Actions
mc: move clockwise
mac: move anticlockwise
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

Formalizing arcs using STRIPS notation

$$\langle G, A, N \rangle$$

where G is $[X_1 = v_1, \ldots, X_n = v_n]$ is an arc if

- $\exists i \; X_i = v_i$ is on the effects list of action A
- $\forall j \; X_j = v_j'$ is not on the effects list for A, where $v_j' \neq v_j$
- N is $\mathit{preconditions}(A) \cup \{X_k = v_k : X_k = v_k \notin \mathit{effects}(A)\}$ and N is consistent in that it does not assign different values to any variable.

Loop detection and multiple-path pruning

A regression planner does not have to visit exactly the same node to prune the search.

- Goal G_1 is simpler (i.e. more general) than goal G_2 if G_1 is a subset of G_2.
  - It is easier to solve $[cs]$ than $[cs, rhc]$.
- If on a path to node N a more specific (i.e. more complex) goal has been found, the path to N can be pruned.

Improving Efficiency

- You can define a heuristic function that estimates how difficult it is to solve the goal from the initial state.
- You can use domain-specific knowledge to remove impossible goals.
  - It is often not obvious from an action description to conclude that an agent can only hold one item at any time.

Comparing forward and regression planners

- Which is more efficient depends on:
  - The branching factor
  - How good the heuristics are
- Forward planning is unconstrained by the goal (except as a source of heuristics).
- Regression planning is unconstrained by the initial state (except as a source of heuristics)
- Regression planning imposes a total ordering on the action sequence. It needs to try all permutations to find out that non of them can be successful.

Planning as a CSP

- Search over planning horizons.
- For each planning horizon, create a CSP constraining possible actions and features
- Also factor actions into action features.

Action Features

- *PUC*: Boolean variable, the agent picks up coffee.
- *DelC*: Boolean variable, the agent delivers coffee.
- *PUM*: Boolean variable, the agent picks up mail.
- *DelM*: Boolean variable, the agent delivers mail.
- *Move*: variable with domain $\{mc, mac, nm\}$ specifies whether the agent moves clockwise, anti-clockwise or doesn't move

CSP Variables

Choose a planning horizon k.

- Create a variable for each state feature and each time from 0 to k.
- Create a variable for each action feature for each time in the range 0 to $k - 1$.

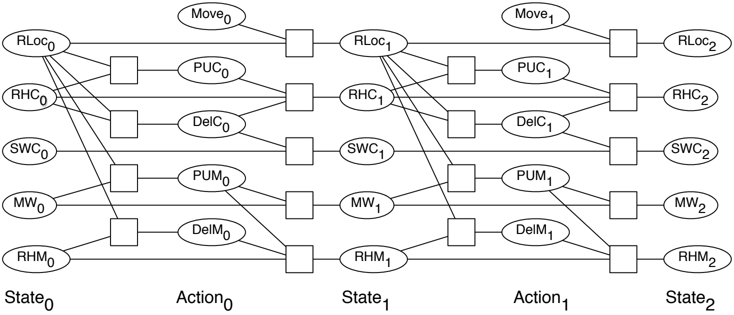
Constraints

- state constraints that are constraints between variables at the same time step.
- precondition constraints between state variables at time t and action variables at time t that specify constraints on what actions are available from a state.
- effect constraints between state variables at time t, action variables at time t and state variables at time $t + 1$.

Constraints

- action constraints that specify which actions cannot co-occur. These are sometimes called mutual exclusion or mutex constraints.
- initial state constraints that are usually domain constraints on the initial state (at time 0).
- goal constraints that constrains the final state to be a state that satisfies the goals that are to be achieved.

CSP for Delivery Robot



$RLoc_i$ — Rob's location
RHC_i — Rob has coffee
SWC_i — Sam wants coffee
MW_i — Mail is waiting
RHM_i — Rob has mail

$Move_i$ — Rob's move action
PUC_i — Rob picks up coffee
$DelC$ — Rob delivers coffee
PUM_i — Rob picks up mail
$DelM_i$ — Rob delivers mail

Effect Constraint

RHC_i	DC_i	PUC_i	RHC_{i+1}
true	true	true	true
true	true	false	false
true	false	true	true
true	false	false	true
false	true	true	false
false	true	false	false
false	false	true	true
false	false	false	false

Choosing a suitable horizon

- The CSP representation assumes a fixed planning horizon, i.e. a fixed number of actions.
- Assuming a do-nothing action, allows CSP solver to find solutions up to a predefined maximum length.
- If the maximum number of steps is unknown, the horizon can be iteratively increased (similar to iterative deepening) until a solution is found.
- This creates a termination problem if no plan exists.

Choosing a suitable horizon

- For stochastic local search, it is possible to consider multiple horizons at once with an upper bound on the their length. The bound can be increased slowly.
- Solving the CSP using arc consistency and search, it sometimes is possible to show that trying a longer plan will not lead to a solution either.