Lecture 13

Planning as CSP

Lecture Overview

- Recap of Lecture 12
- Planning as CSP
  - Details on CSP representation -

  Solving the CSP planning problem
- Intro to Logic (time permitting)

Course Overview

Problem Type Arc **Environment** Stochastic

Representation Reasoning Technique

Deterministic

Consistency

Constraint Satisfaction ***Vars + Constraints*** Search

Static ***Belief Nets***

Query ***Logics***

Search

Variable

Elimination

Sequential ***STRIPS*** ***Decision Nets*** Variable

Planning Search Elimination

We'll focus ***Markov Processes*** Value

on Planning Iteration

Planning Problem

- Goal
- Description of states of the world

- Description of available actions => when each action can be applied and what its effects are

- Planning: build a sequence of actions that, if executed, takes the agent from the current state to a state that achieves the goal

Standard Search vs. Specific R&R systems

- Constraint Satisfaction (Problems):
- State: assignments of values to a subset of the variables
- Successor function: assign values to a "free" variable
- Goal test: all variables assigned a value and all constraints satisfied?
- Solution: possible world that satisfies the constraints

- Heuristic function: none (all solutions at the same distance from start)

- Planning:
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function
  - Inference
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function

Key Idea of Planning

- Open-up the representation of states, goals and actions
  - Both states and goals as set of features
  - Actions as preconditions and effects defined on state features

⇓

- agent can reason more deliberately about which actions to consider to achieve its goals.

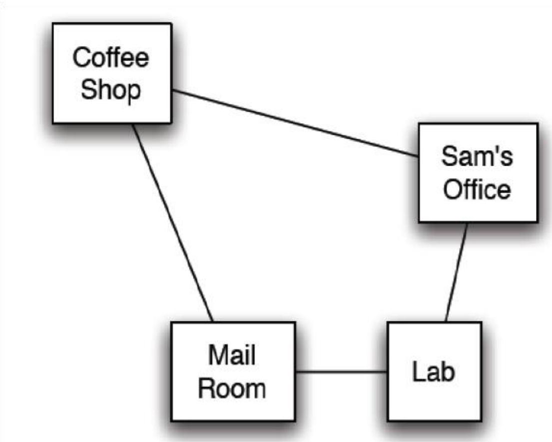
Delivery Robot Example: features

- RLoc - Rob's location
- Domain: {coffee shop, Sam's office, mail room, lab} short {cs, off, mr, lab}
- RHC - Rob has coffee
- Domain: {true, false}. Alternatively notation for RHC = T/F: rhcindicates that Rob has coffee, and that Rob doesn't'have coffee*rhc*
- SWC - Sam wants coffee {true, false}
- MW - Mail is waiting {true, false}

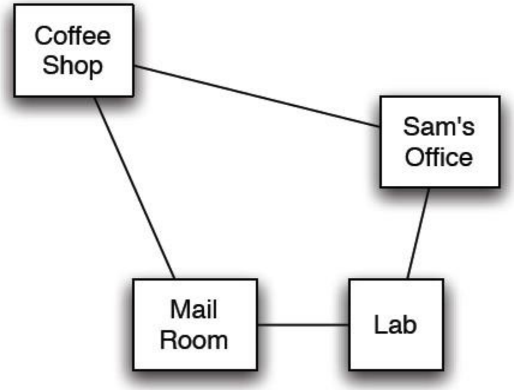
-

- RHM - Rob has mail {true, false}
- An example state is

$$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$$



Rob is in the lab, it does not have coffee, Sam wants coffee, there is no mail waiting and Rob has mail



Delivery Robot Example: Actions

The robot's actions are:

puc - Rob picks up coffee

- must be at the coffee shop and not have coffee

Preconditions for action application

delC - Rob delivers coffee

- must be at the office, and must have coffee

pum - Rob picks up mail

- must be in the mail room, and mail must be waiting

delM - Rob delivers mail

- must be at the office and have mail *move* - Rob's move

actions – there are 8 of them

- move clockwise (*mc*-*x*), move anti-clockwise (*mac*-*x*) from

location x (where x can be any of the 4 rooms)

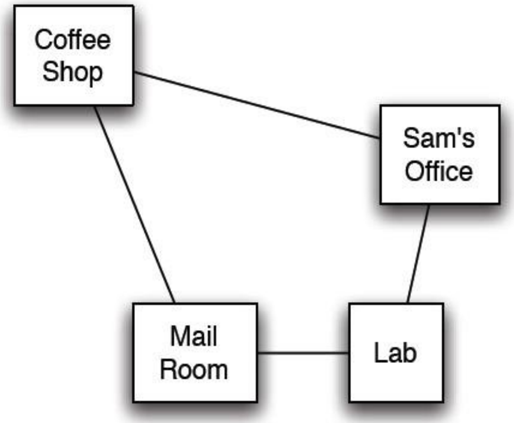
- must be in location x

Modeling actions for planning

- The key to sophisticated planning is modeling actions
- Leverage a feature-based representation:
- Model when actions are possible, in terms of the values of the features in the current state

- Model state transitions caused by actions in terms of changes in specific features

STRIPS actions: Example



STRIPS representation of the action pick up coffee, puc:

- preconditionsLoc= cs and RHC= F
- effectsRHC= T

cs = coffee shop

off = Sam's office

mr = mail rom

STRIPS representation of the action deliver coffee, Del :

- preconditions Loc= off and RHC= T
- effects RHC = T and SWC = F

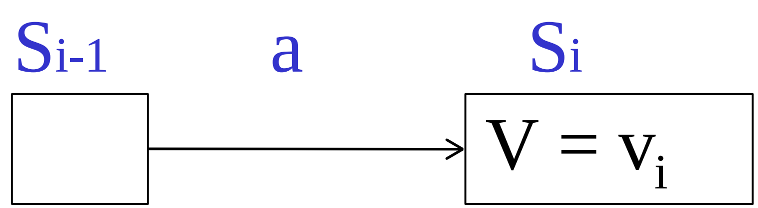
Note in this domain Sam doesn't have to want coffee for Rob to deliver it; one way or another, Sam doesn't want coffee after delivery.

STRIPS Actions (cont')

The STRIPS assumption:

all features not explicitly changed by an action stay unchanged

- So if the feature **V** has value $\mathbf{v_i}$ in state $\mathbf{S_i}$, after action **a** has been performed,
- what can we conclude about **a** and/or the state of the world S_{i-1} immediately preceding the execution of **a**?



C. At least one of A and B

Solving planning problems

- STRIPS lends itself to solve planning problems either

  - $\boxed{\text{As pure search problems}}$
- As CSP problems

- We will look at one technique for each approach

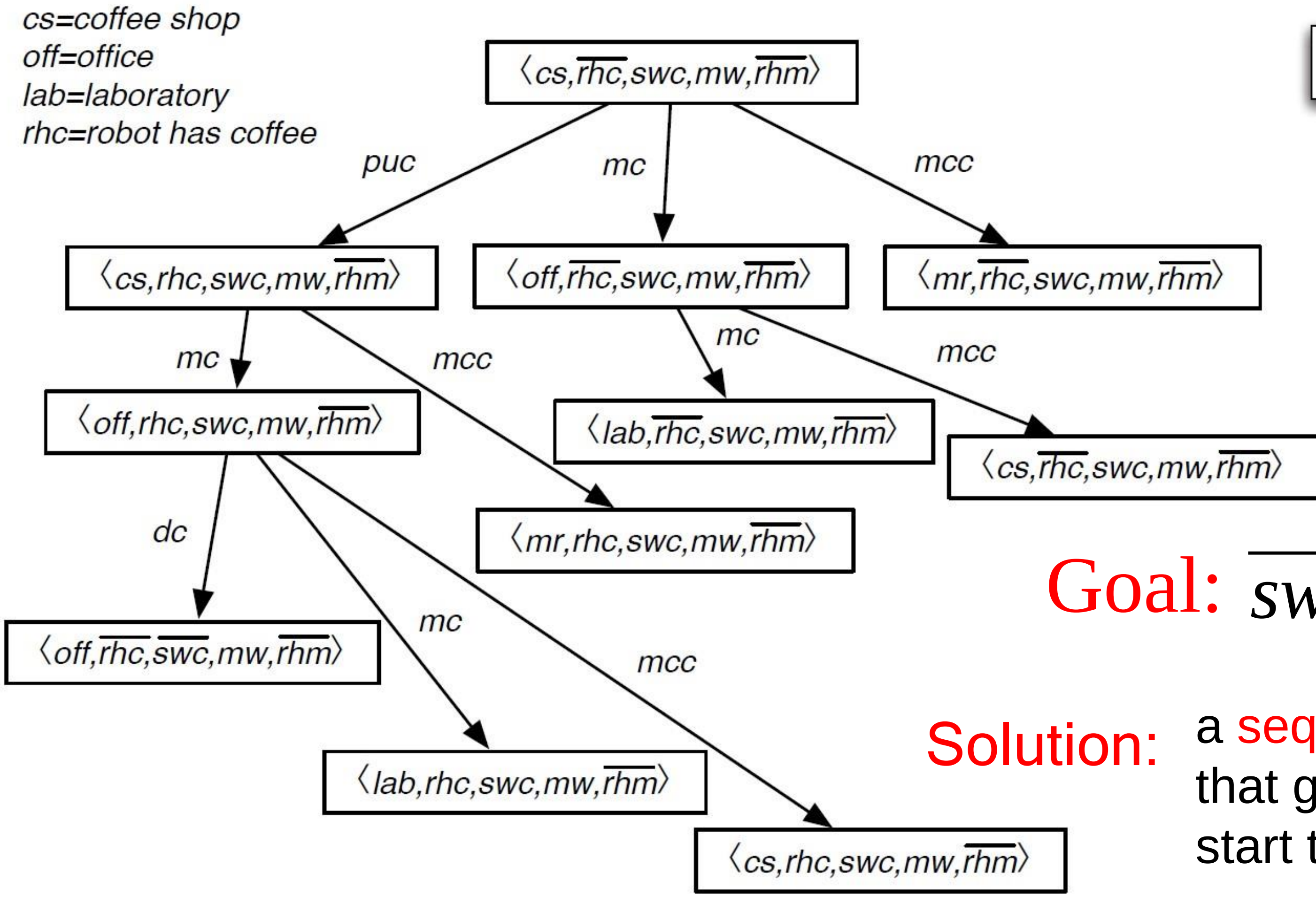
Forward planning

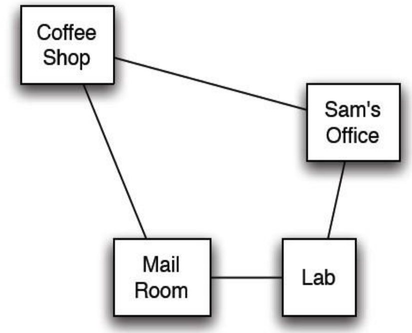
- To find a plan, a solution : search in the state-space graph
- The states are the possible worlds
  - ✓ full assignments of values to features
- The arcs from a state s represent all the actions that are possible in state s
- A plan is a path from the state representing the initial state to a state that satisfies the goal

Which actions a are possible in a state s?

B. Those where a’s preconditions are satisfied in s

Example for state space graph





Goal: $\overline{swc}$

Solution: a sequence of actions that gets us from the start to a goal

What is a solution to this planning problem?

D (puc, mc, dc)

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

- State: assignments of values to a subset of the variables
- Successor function: assign values to a “free” variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: *none (all solutions at the same distance from start)*

Planning :

- State: full assignment of values to features
- Successor function: states reachable by applying actions with preconditions satisfied in the current state
- Goal test: partial assignment of values to features
- Solution: a sequence of actions
- Heuristic function

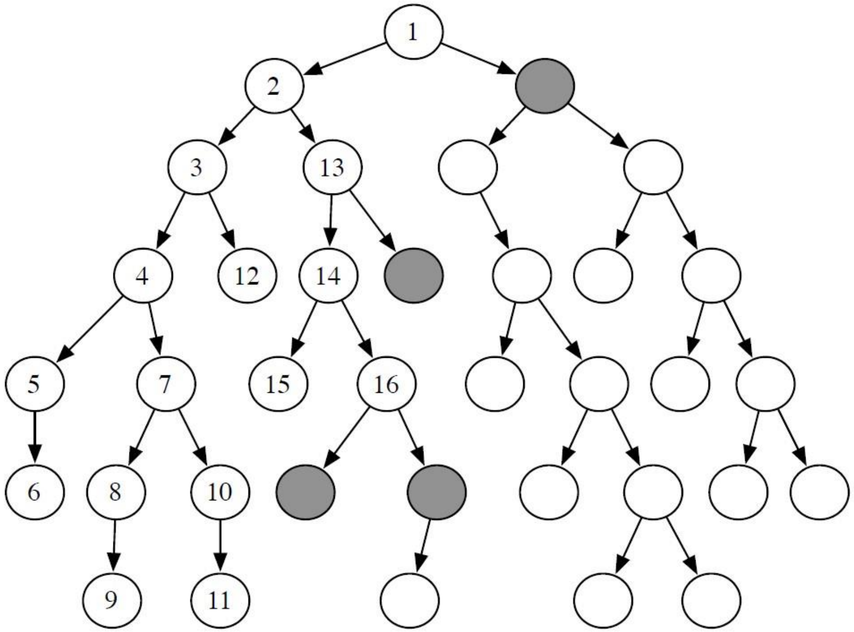
Inference

- State

- Successor function
- Goal test
- Solution
- Heuristic function

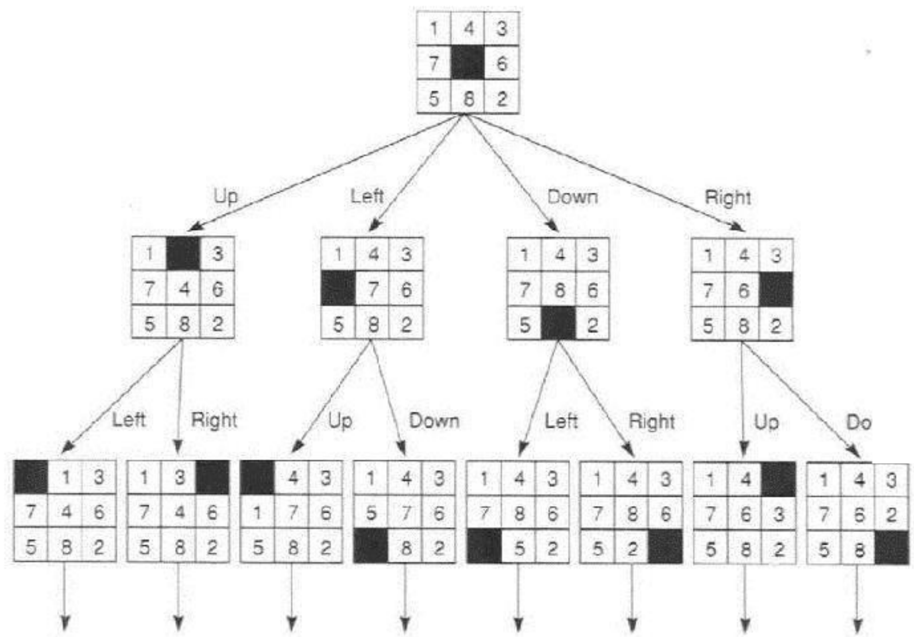
Search Spaces Seen So Far

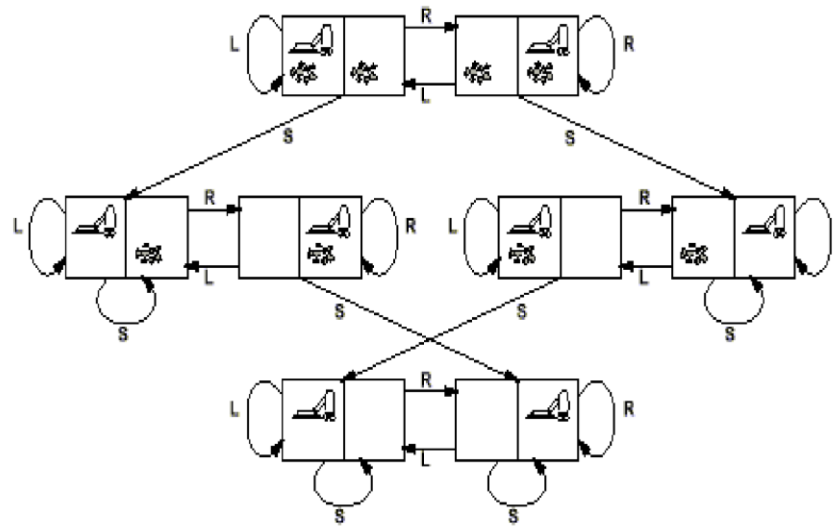
For generic search algorithms



Search Spaces Seen so Far

For generic search algorithms





- Actions - left, right, suck
  - Successor states in the graph describe the effect of each action applied to a given state

Actions: blank moves left, right, up down

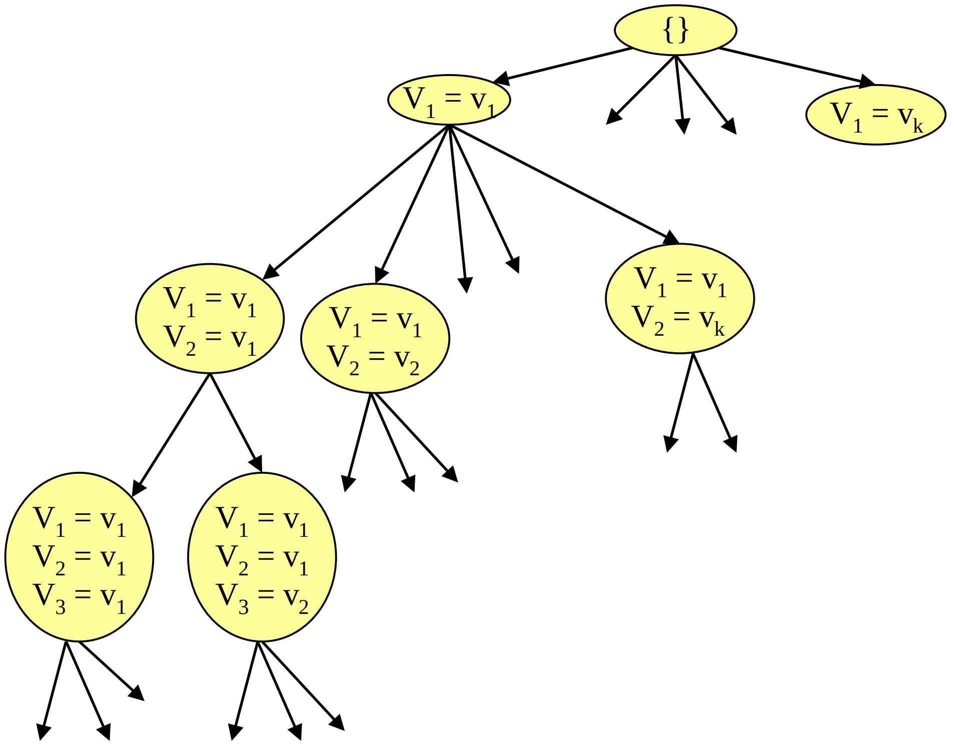
- Possible Goal - no dirt Goal: configuration with numbers in right sequence

Search Spaces Seen so Far

For solving CSP problems with Search

- Constraint Satisfaction (Problems):
- State: assignments of values to a subset of the variables

- Successor function: assign values to a "free" variable
- Goal test: all variables assigned a value and all constraints satisfied?
- Solution: possible world that satisfies the constraints
- Heuristic function: none (all solutions at the same distance from start)



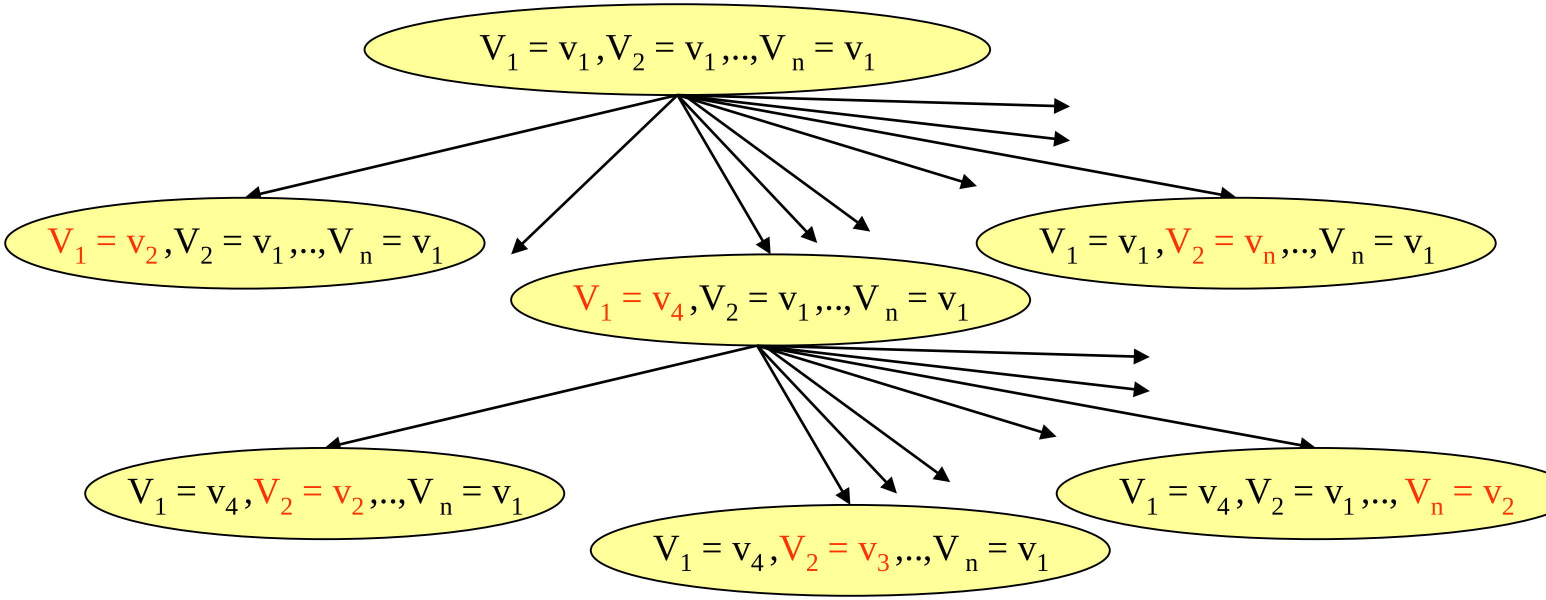
Search Spaces Seen so Far

For solving CSP problems with Stochastic Local Search

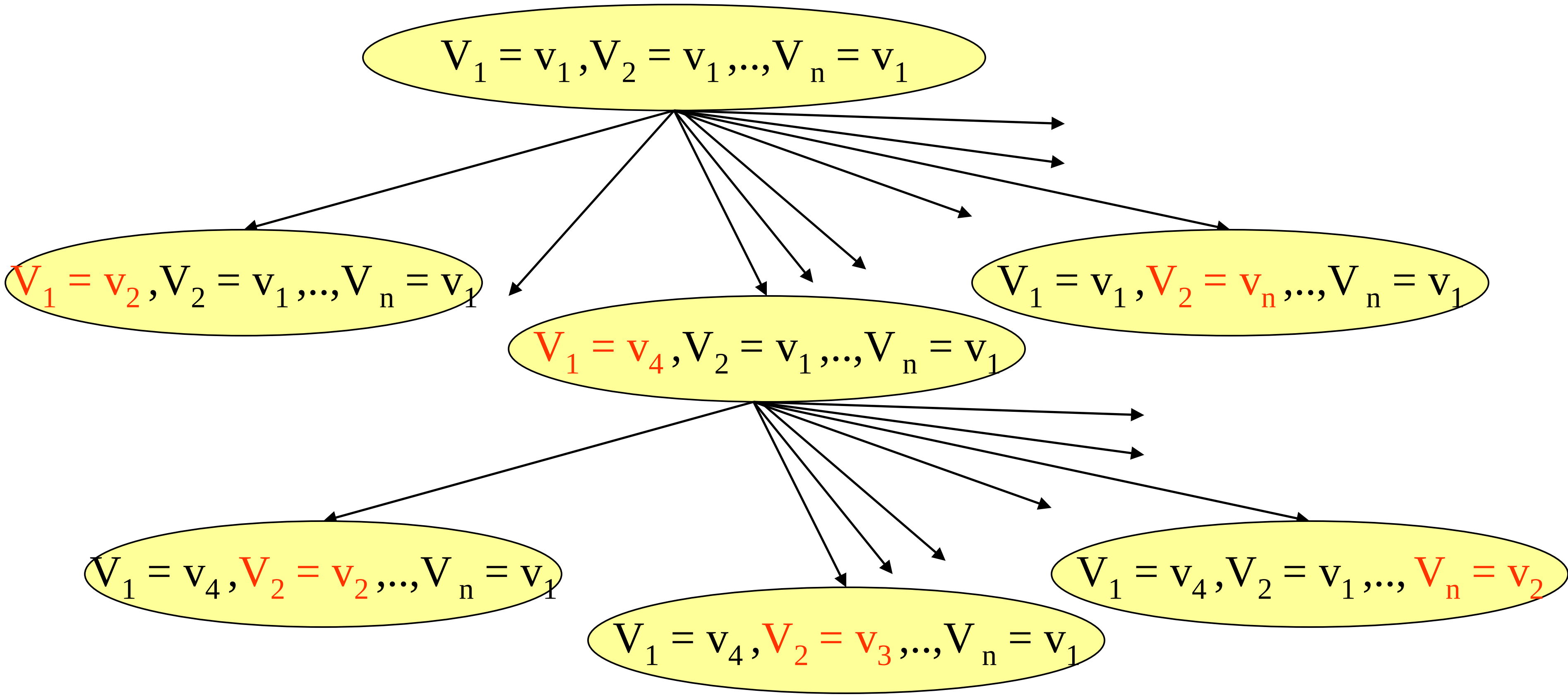
- Given the set of variables $\{V_1 \ldots., V_n\}$, each with domain $Dom(V_i)$
- States are full assignments of values to variables

- The start node is any assignment $\{V_1 / v_1,\ldots,V_n / v_n\}$.
- The neighbors of node with assignment

  $A= \{V_1 / v_1,\ldots,V_n / v_n\}$ are nodes with assignments that differ from A for one value only



Search Space



- Only the current node is kept in memory at each step.
- Very different from the systematic tree search approaches we have seen so far!

- Local search does NOT backtrack!

Anything else

Another formulation of CSP as search

Arc consistency with domain splitting

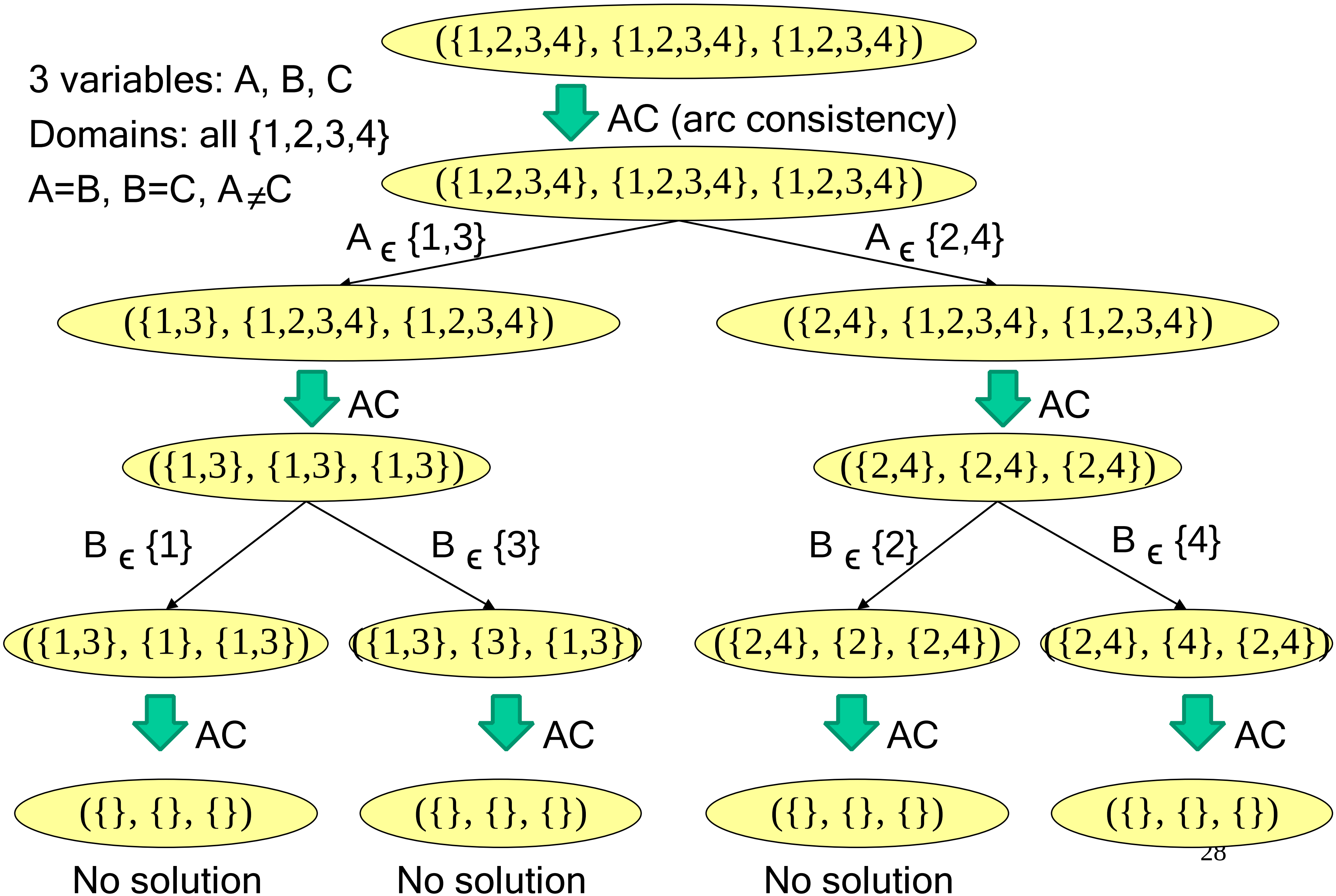
- States: vector $(D(V_1), \ldots, D(V_n))$ of remaining domains, with $D(V_i) \subseteq dom(V_i)$ for each V_i
- Start state: vector of original domains $(dom(V_1), \ldots, dom(V_n))$
- Successor function:
- reduce one of the domains + run arc consistency
- Goal state: vector of unary domains that satisfies all constraints
- That is, only one value left for each variable
- The assignment of each variable to its single value is a model
- Solution: that assignment

Arc consistency + domain splitting: example

3 variables: A, B, C

Domains: all {1,2,3,4}

A=B, B=C, A$\neq$C



Forward Planning

- Any of the search algorithms we have seen can be used in Forward Planning
- Problem?
- Complexity is defined by the branching factor, e.g.…

  Number of applicable actions to a state

- Can be very large

- Solution?

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

- State: assignments of values to a subset of the variables
- Successor function: assign values to a "free" variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: *none (all solutions at the same distance from start)* Planning :
- State: full assignment of values to features
- Successor function: states reachable by applying actions with preconditions satisfied in the current state
- Goal test: partial assignment of values to features
- Solution: a sequence of actions
- Heuristic function

Inference

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Heuristics for Forward Planning

Not in textbook, but you can see details in Russel&Norvig, 10.3.2

- Heuristic function: estimate of the distance from a state to the goal
- In planning this distance

is the……………….B. # of actions needed to get from s to the goal

- Finding a good heuristics is what makes forward planning feasible in practice
- Factored representation of states and actions allows for definition of domain-independent heuristics
- We will look at one example of such domain-independent heuristic that has proven to be quite successful in practice

Heuristics for Forward Planning: ignore delete-list

- One strategy to find an admissible heuristics is
- to relax the original problem
- One way : remove all the effects that make a variable = F. Action aeffects (B=F, C=T)
- If we find the path from the initial state to the goal using this relaxed version of the actions:
- the length of the solution is an underestimate of the actual solution length
- Why? In the original problem, one action (e.g. aabove) might undo an already achieved goal (e.g. by a1 below). It would have to be achieved

again

a1 B = T

A = TA = T

Goal

B = F

C = T

S_0 B = F a B = T

C = FC = T

Heuristics for Forward Planning: ignore delete-list

But how do we compute the actual heuristics values for ignore delete-list?

- To compute $h(s_i)$, run forward planner with
- s_i as start state

- Same goal as original problem
- Actions without "delete list"

- Often fast enough to be worthwhile
  ✓Planning is PSPACE-hard (that's really hard, includes NP-hard)
  ✓Without delete lists: often very fast

Can you think of another way of deriving a domain-independent admissible heuristic for planning?

- Let's stay in the robot domain • But say our robot has to bring coffee to Bob, Sue, and Steve:

- G = {bob_has_coffee, sue_has_coffee, steve_has_coffee}
- They all sit in different offices

A. Count Number of satisfied sub-goals

B. Count Number of unsatisfied sub-goals

C. Plan by ignoring action preconditions

Can you think of another way of deriving a domain independent admissible heuristic for planning?

- Let's stay in the robot domain • But say our robot has to bring coffee to Bob, Sue, and Steve:
- G = {bob_has_coffee, sue_has_coffee, steve_has_coffee}

D. None of the above

- They all sit in different offices

- BOTH B. Count Number of unsatisfied sub-goals

C. Plan by ignoring action preconditions

✓in this domain, and for this goal, because one cannot achieve more than one subgoal with a single action

✓In general, only C; B might be an overestimate of the actual cost when actions can achieve multiple subgoals Slide 33

Example

- Let's stay in the robot domain • But say our robot has to bring coffee to Bob, Sue, and Steve:
- G = {bob_has_coffee, sue_has_coffee, steve_has_coffee}

- They all sit in different offices

- Admissible heuristic: ignore preconditions:
- Can simply apply “Deliver Coffee(person)” action for each person (since there is no need to have coffee to deliver it)

Solving planning problems

- STRIPS lends itself to solve planning problems either

- As pure search problems

- As CSP problems

- We will look at one technique for each approach

Lecture Overview

- Recap of Lecture 12
- Planning as CSP

- Details on CSP representation -

Solving the CSP planning problem

- Intro to Logic (time permitting)

Course Overview

Problem Type

Arc **Environment** Stochastic

Representation Reasoning Technique

Deterministic

Consistency

Constraint Satisfaction ***Vars + Constraints*** Search

Static ***Belief Nets***

Query ***Logics***

Search

Variable

Elimination

Sequential ***STRIPS*** ***Decision Nets*** Variable

Planning Search Elimination

Markov Processes

Value

Iteration

Planning as a CSP

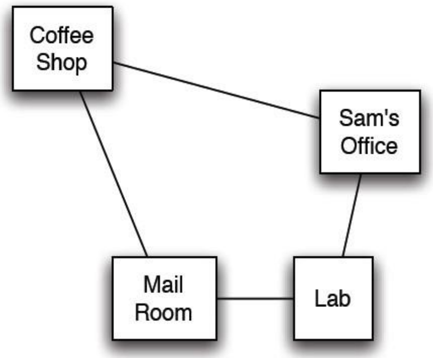
- We simply reformulate a STRIPS model as a set of variables and constraints
- Give it a try: please work in groups of two or three for a few minutes and try to define what would you chose as

- Variables
- Constraints
- Use the Rob Delivery World as a leading example

What will be the CSP variables and constraints?

- Features change over time
- Might need more than one CSP variable per feature

- Initial state constraints
- Goal state constraints



- STRIPS example actions
- 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 $\overline{swc}$

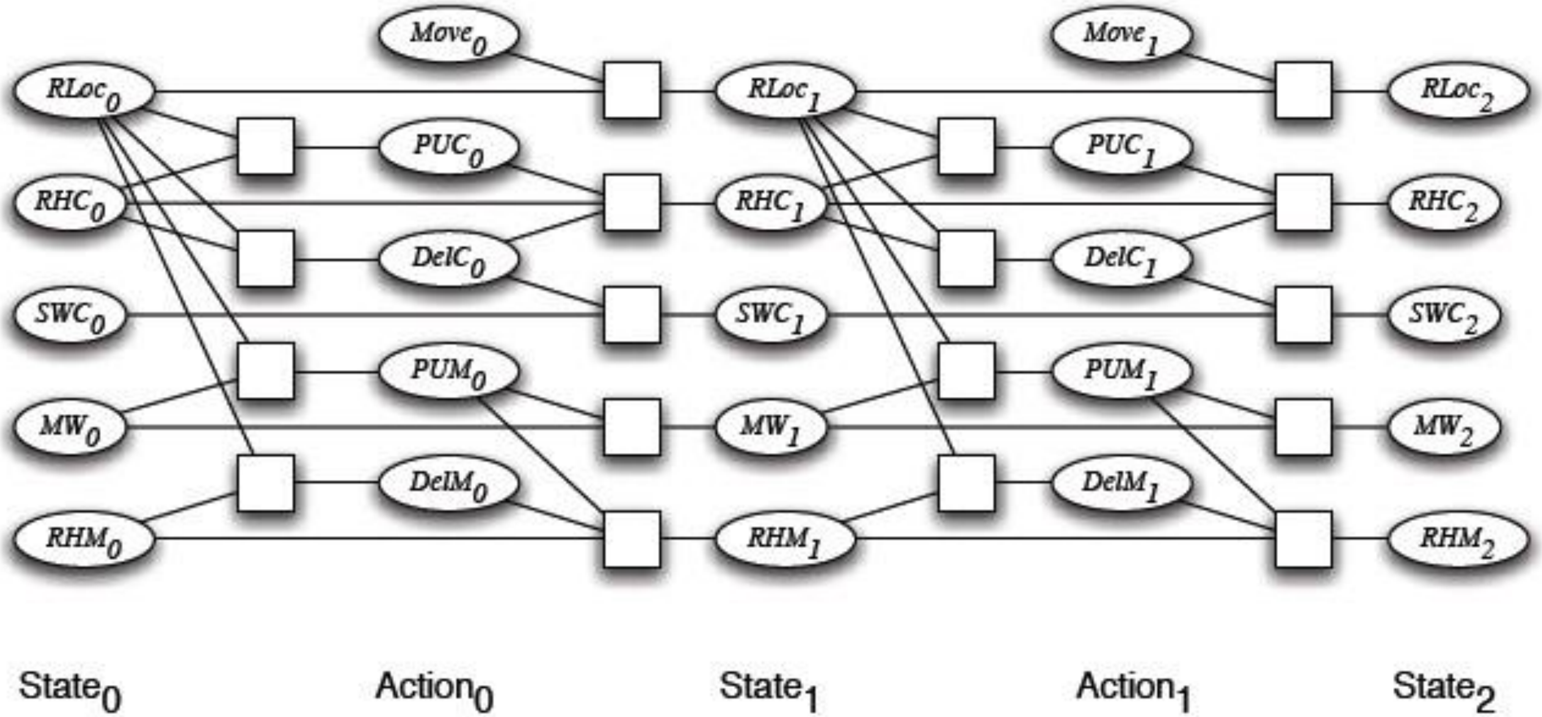
Have to capture these conditions as constraints

Planning as a CSP: General Idea

- Action preconditions and effects are virtually constraints between
- the action,
- the states in which it can be applied
- the states that it can generate
- Thus, we can make both states and actions into the variables of our CSP formulations
- However, constraints based on action preconditions and effects relate to states at a given time t, the corresponding valid actions and the resulting states at t +1
- need to have as many state and action variables as there are planning steps

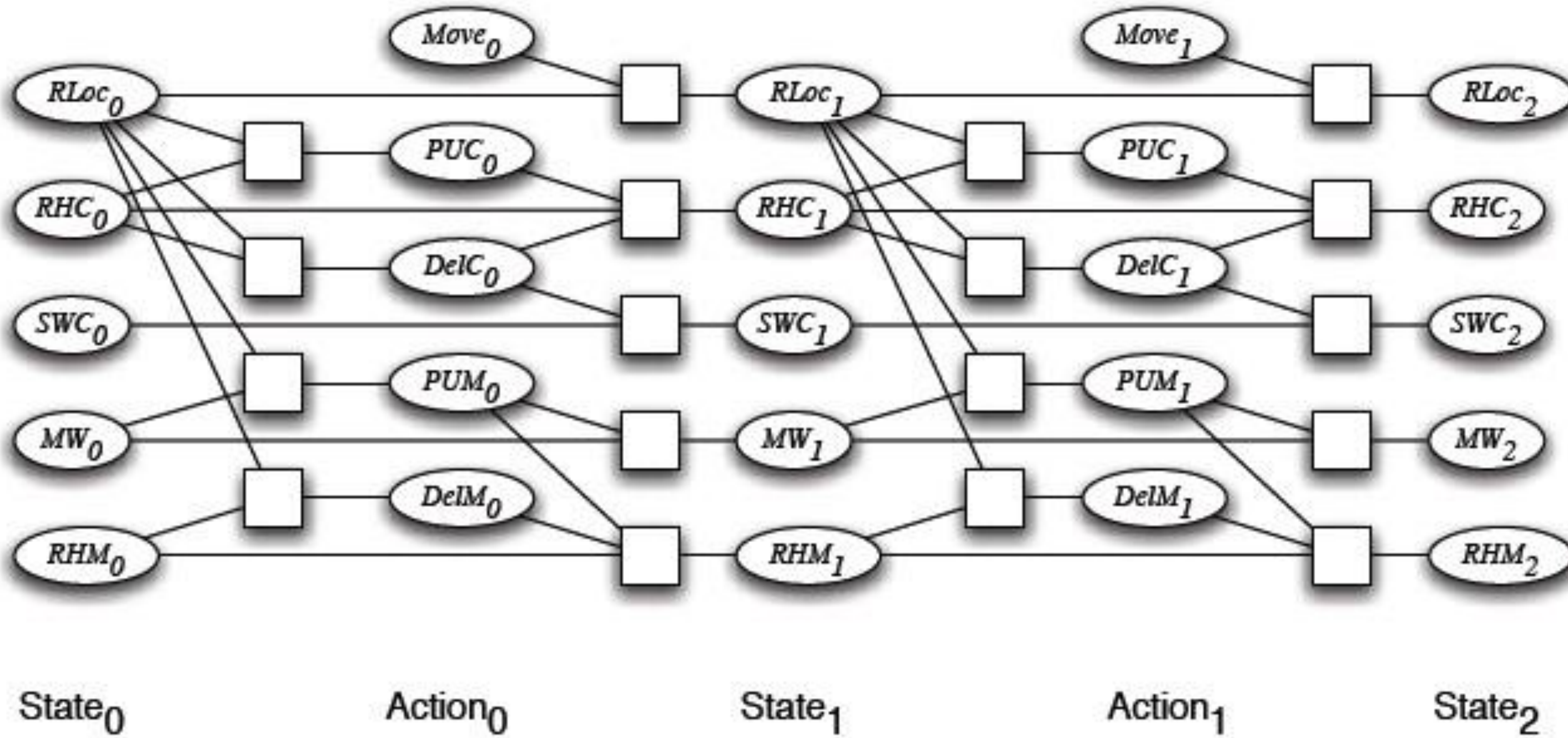
Planning as a CSP: General Idea

- Both features and actions are CSP variables
- Action preconditions and effects are constraints among
- the action,
- the states in which it can be applied
- the states that it can generate



Planning as a CSP: General Idea

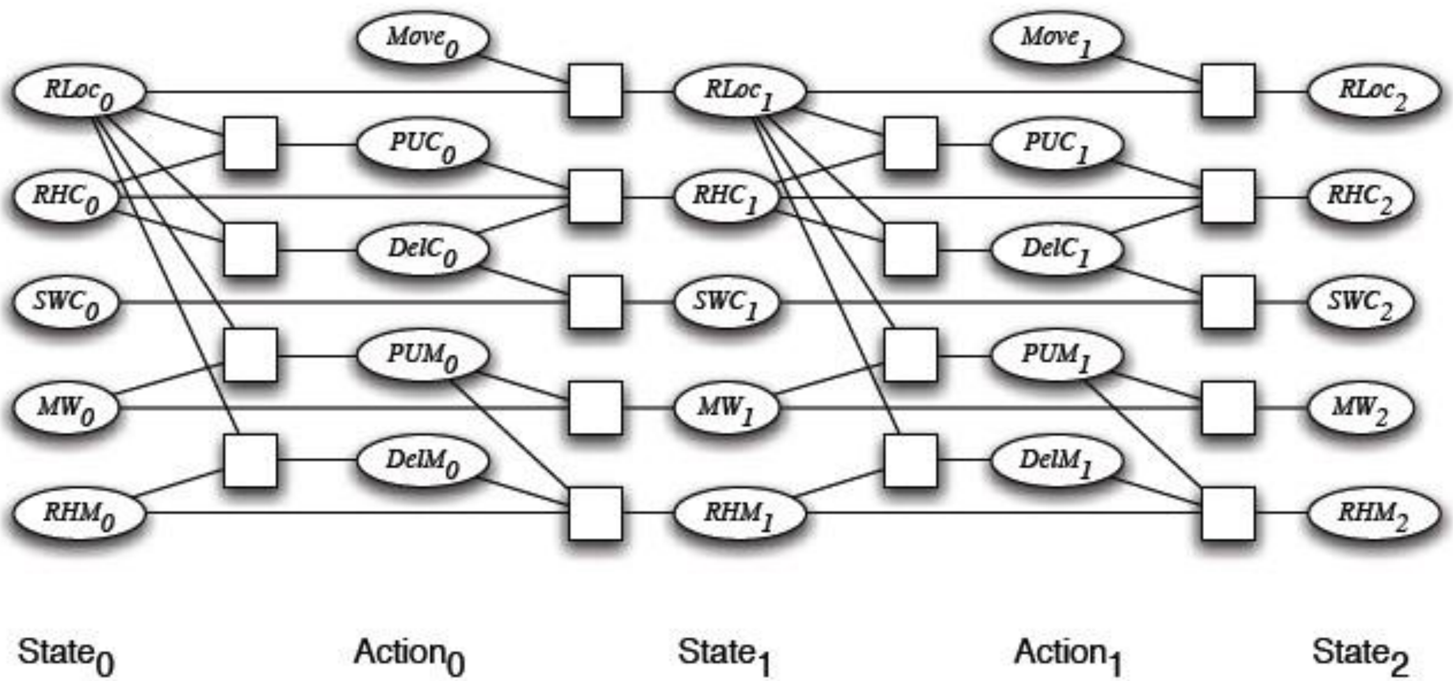
- These action constraints relate to states at a given time t, the corresponding valid actions and the resulting states at t +1
- we need to have as many state and action variables as we have planning steps



Planning as a CSP: Variables

- We need to 'unroll the plan' for a fixed number of steps: this is called the horizon k
- To do this with a horizon of k:

- construct a CSP variable for each STRIPS state variable at each time step from 0 to k
- construct a Boolean CSP variable for each STRIPS action at each time step from 0 to k - 1.



Lecture Overview

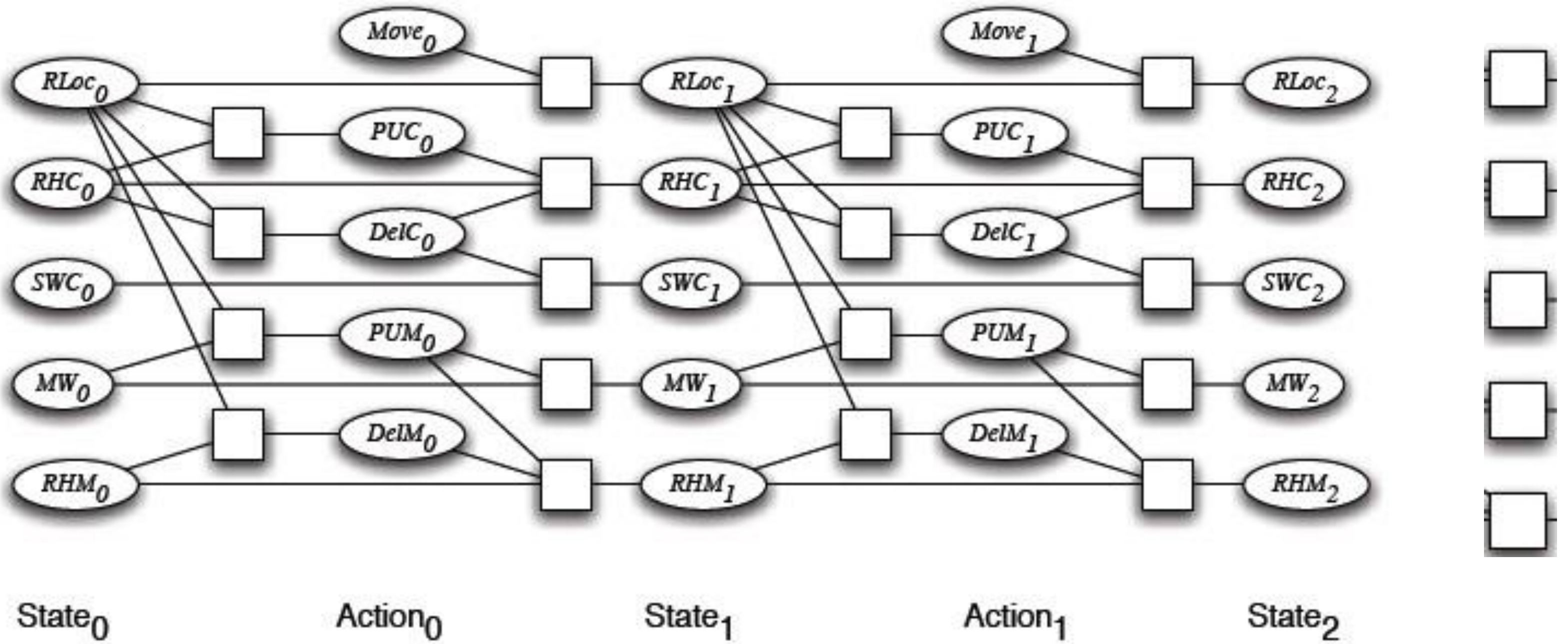
- Recap of Lecture 12
- Planning as CSP

  - Details on CSP representation - Solving the CSP planning problem

- Intro to Logic (time permitting)

Initial State(s) and Goal(s)

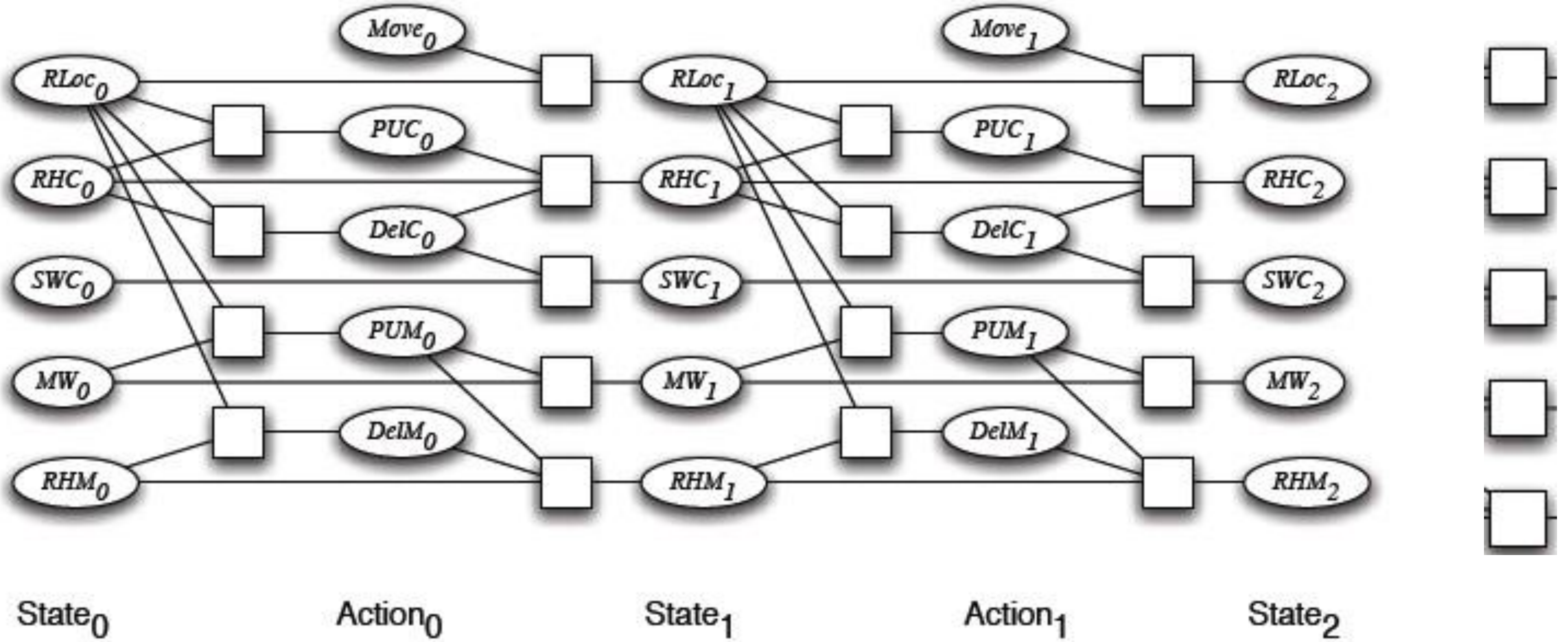
- How can we represent the initial state(s) and the goal(s) with this representation? E.g.,
  - ✓Initial state with Sam wanting coffee and Rob at the coffee shop, with no coffee
  - ✓Goal: Sam does not want coffee



Initial State(s) and Goal(s)

- How can we represent the initial state(s) and the goal(s) with this representation? E.g.,
  - ✓Initial state with Sam wanting coffee and Rob at the coffee shop, with no coffee

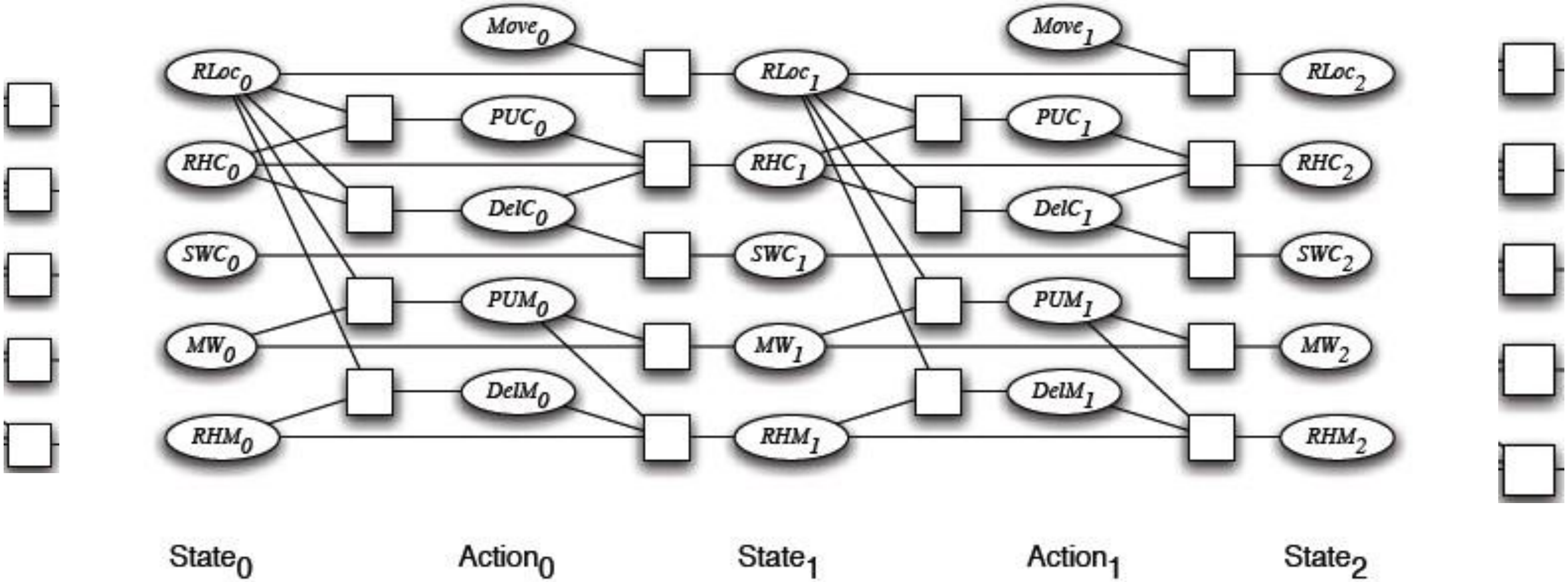
✓Goal: Sam does not want coffee



Initial and Goal Constraints

- initial state constraints: unary constraints on the values of the state variables at time 0

- goal constraints: unary constraints on the values of the state variables at time k



CSP Planning: Precondition Constraints

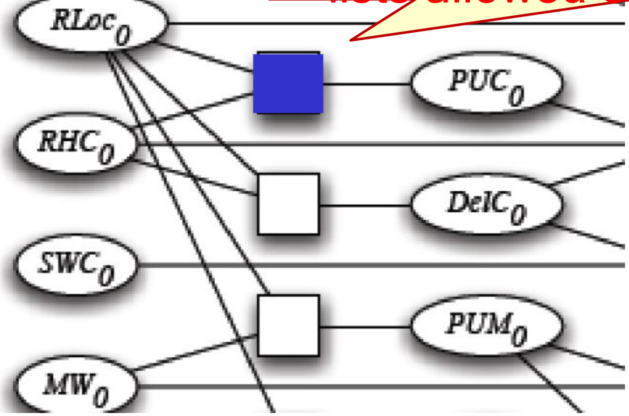
precondition constraints

- between state variables at time t and action variables at time t
- specify when actions may be taken. E.g.,
  - ✓robot can only pick up coffee when Loc=cs (coffee shop) and

RHC = false (don't have coffee already)

$RLoc_0$	RHC_0	PUC_0
cs	T	
cs	F	
cs	F	
mr	T/F	
lab	T/F	

Truth table for this constraint:
lists allowed combinations of values



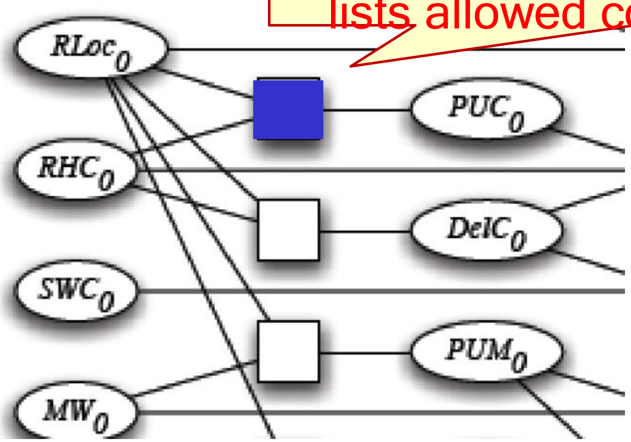
off	T/F	

CSP Planning: Precondition Constraints

precondition constraints

- between state variables at time t and action variables at time t

Truth table for this constraint:
lists allowed combinations of values



RLoc$_0$	RHC$_0$	PUC$_0$
cs	T	F
cs	F	T
cs	F	F
mr	*	F
lab	*	F
off	*	F

Need to allow for the option of *not* taking an action even when it is valid

- specify when actions may be taken. E.g.,
  - ✓robot can only pick up coffee when Loc=cs (coffee shop) and RHC = false (don't have coffee already)

Learning Goals for Planning

- STRIPS
- Represent a planning problem with the STRIPS representation
- Explain the STRIPS assumption

- Forward planning
- Solve a planning problem by search (forward planning). Specify states, successor function, goal test and solution

INCLUDED IN THE MIDTERM UP TO HERE

- Construct and justify a heuristic function for forward planning

- CSP planning
- Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)

- Variables 51