CPSC 322: Introduction to Artificial Intelligence

Planning: Representation and Forward Search

Textbook reference: [6.1, 6.2]

Instructor: Varada Kolhatkar
University of British Columbia

Credit: These slides are adapted from the slides of the previous offerings of the course. Thanks to all instructors for creating and improving the teaching material and making it available!

Announcements

- Final exam scheduled: **Dec 9 at 7:00pm**

- Reminder: Midterm next Friday, **Oct 25 from 6pm to 7pm** in Woodward 2

- Midterm **practice questions** are available on Piazza.

- I will be holding **extra office hours/review session** on Thursday during class time (5 to 6:30pm) in our usual classroom (LSK 201).

Midterm

Included in midterm:

- Everything till (and including) lecture 13
- Planning is **not** included in midterm

Structure of midterm:

- Multiple choice or True/False questions
- Short answer questions
- Problem solving questions

Midterm tips

- Make sure you understand the purpose and basic ideas of the different algorithms we have learned so far.

- Go through lecture notes and the lecture learning outcomes and make sure you can do things that are expected of you.

- Make use of extra office hours/review session.

Midterm tips

- Make sure you understand the purpose and basic ideas of the different algorithms we have learned so far.

- Go through lecture notes and the lecture learning outcomes and make sure you can do things that are expected of you.

- Make use of extra office hours/review session.

We like answers that are: **correct**, **easy to read**, and **as short as possible while including important information**. We may penalize answers containing irrelevant information.

Lecture outline

- Recap stochastic local search 👈
- Planning introduction
- Planning example
- STRIPS: a feature-based representation
- Forward planning
- Summary and wrap up

Recap: Simulated annealing (pair-share)

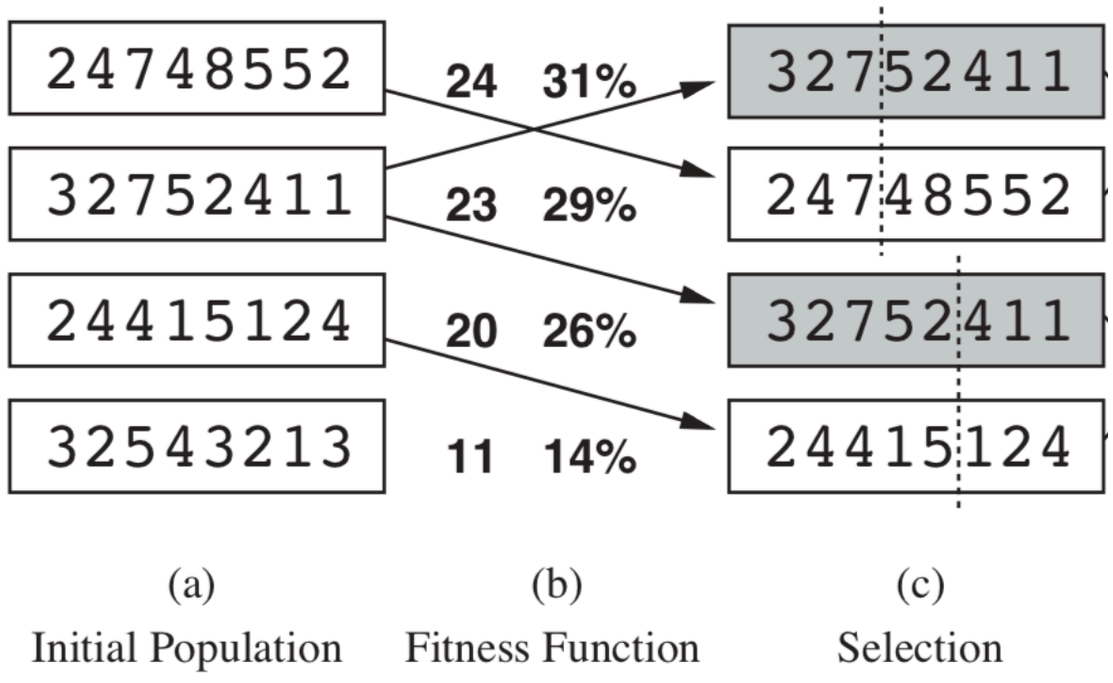
True or False?

- Simulated annealing always accepts the neighbour n' that improves the situation (if there is one).
- Higher temperature means it's less likely to pick a worse neighbour n'.

Recap: Genetic algorithms (pair-share)

- How do we get to 31% from 24?
  $24/(24 + 23 + 20 + 11) = 0.31$

- Why is the individual 32752411 selected twice in (c)?
  Since it's selected randomly from a probability distribution and 32752411 has a high probability of 0.29



A rough CPSC 322 overview

Representation and reasoning

Problem

Environment

Deterministic | Stochastic

Static | Sequential

Constraint satisfaction | Query | Planning

Arc consistency Variables + constraints — Search	
Logics — Search	Belief networks — Variable elimination
STRIPS — Search	Decision networks — Variable elimination Markov decision processes — Value iteration

Today: Learning outcomes

From this lecture, students are expected to be able to:

- Represent a planning problem with the STRIPS representation

- Explain the STRIPS assumption

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

Lecture outline

- Recap stochastic local search
- Planning introduction 👉
- Planning example
- STRIPS: a feature-based representation
- Forward planning
- Summary and wrap up

Why planning?

- Intelligent agents must operate in the world.

- They are not simply passive reasoners (Knowledge Representation, reasoning under uncertainty) or problem solvers (Search), they must also act in the world

- We want intelligent agents to act in "intelligent ways", taking purposeful actions, predicting the expected effect of such actions, composing actions together to achieve complex goals.

Source: http://www.cs.toronto.edu/~sheila/384/w14/Lectures/csc384w14-Lecture-06-Planning.pdf

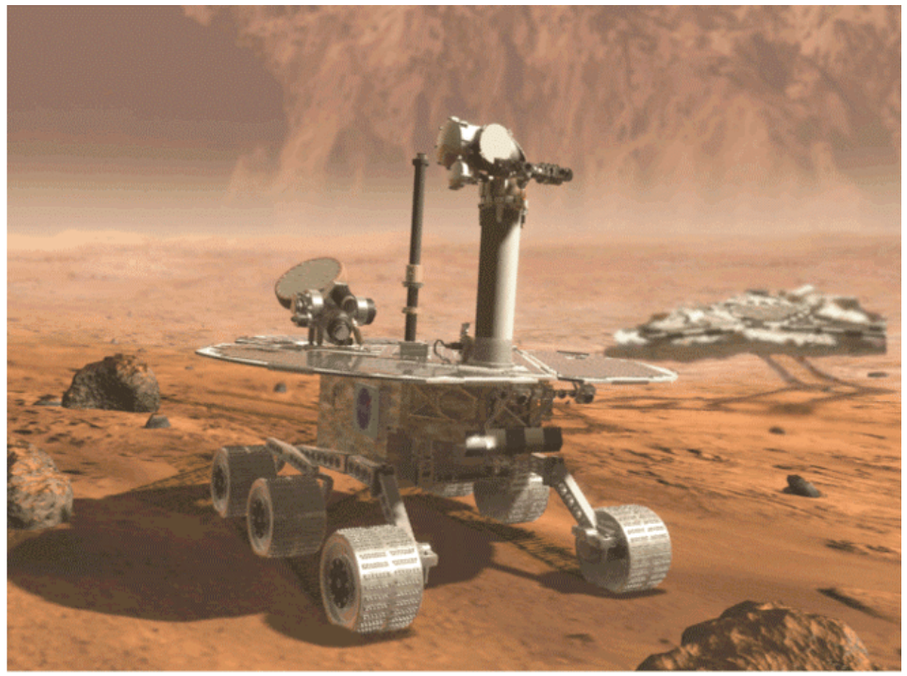
Planning: Autonomous agents

Therapeutic robots

A retirement home resident cuddles a Paro. The FDA considers the faux harp seal a Class II medical device.

© KIM KYUNG-HOON/REUTERS/CORBIS

Mars exploration rover



Coffee delivery drone

Planning: Example

What is its **purpose**?

What **actions** could it take?

Are there any **restrictions** on its actions?

What kind of **plan** would he need to be able to make in order to carry out its function?

Coffee delivery drone

Planning

With CSPs, we looked for solutions to essentially **atemporal** problems

- find a single variable assignment (state) that satisfies all of our constraints
- did not care about the path leading to that state

Planning

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 we 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

Planning

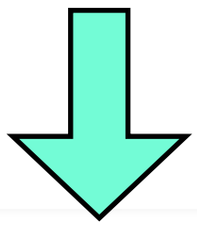
Key idea: "Open up" the representation of states, goals and actions

- States and goals: as features (variable assignments), as in CSP
- Actions: as **preconditions** and **effects** defined on features

Planning

Key idea: "Open up" the representation of states, goals and actions

- States and goals: as features (variable assignments), as in CSP
- Actions: as **preconditions** and **effects** defined on features



Agent can reason more deliberately about what actions to consider to achieve its goals.

Contrast this to simple graph search

How did we represent the problem in graph search?

- States, start states, goal states, and successor function
- Successor function: when applying action a in state s, you end up in s'

Contrast this to simple graph search

How did we represent the problem in graph search?

- We used a 'flat' state-based representation
  - There's no sense in which we can say that states a and b are more similar than states a and z (they're just nodes in a graph)
- Thus, we can't represent the successor function any more compactly than a **tabular representation**

Starting state	Action	Resulting state
⋮	⋮	⋮

Problems with the tabular representation

- Usually **too many states** for a tabular representation to be feasible

- Small changes to the model can mean **big changes** for the representation. E.g., if we add another variable, all the states would change

- There may be **structure and regularity** to the states or to the actions

  - No way to capture this with a tabular representation

Feature-based representation

- Features helped us to represent CSPs more compactly than states could

- The main idea: factor states into joint variable assignments

- Each constraint only needed to mention the variables it constraints

- That enabled efficient constraint propagation: arc consistency

No way to do this in flat state-based representation

Feature-based representation

Want to use similar idea when searching for a sequence of actions that brings us from a start state to a goal state

Main idea: compact, rich representation and efficient reasoning

Solving planning problems

We will be looking at two ways to solve planning problems

- **Forward planning**
  (planning as state-space search)
- Planning as CSP (next lecture)

Planning as search

How to select and organize a sequence of actions to achieve a given goal.

State: Agent is in a possible world (full assignments to a set of variables/features)

Goal: Agent wants to be in a possible world where some variables are given specific values

Planning as search

How to select and organize a sequence of actions to achieve a given goal.

State: Agent is in a possible world (full assignments to a set of variables/features)

Goal: Agent wants to be in a possible world where some variables are given specific values

Example:

Variables: A, B, C

Domains: {True (T), False (F)}

Example state: $\begin{bmatrix} A = T \\ B = F \\ C = T \end{bmatrix}$

Goal: $[A = T \quad B = T \quad C = F]$
or $[A = T \quad B = F \quad C = F]$

Planning as search: Successor function and solution

Actions: Take the agent from one state to another

Example:
Variables: A, B, C
Domains: {True (T), False (F)}
Goal: $[A = T \quad C = F]$

$$\begin{bmatrix} A = T \\ B = F \\ C = T \end{bmatrix} \xrightarrow{a_1} \begin{bmatrix} A = F \\ B = F \\ C = T \end{bmatrix} \xrightarrow{a_2} \begin{bmatrix} A = T \\ B = F \\ C = F \end{bmatrix}$$

Solution: sequence of actions that when performed will take the agent from the current state to a goal state

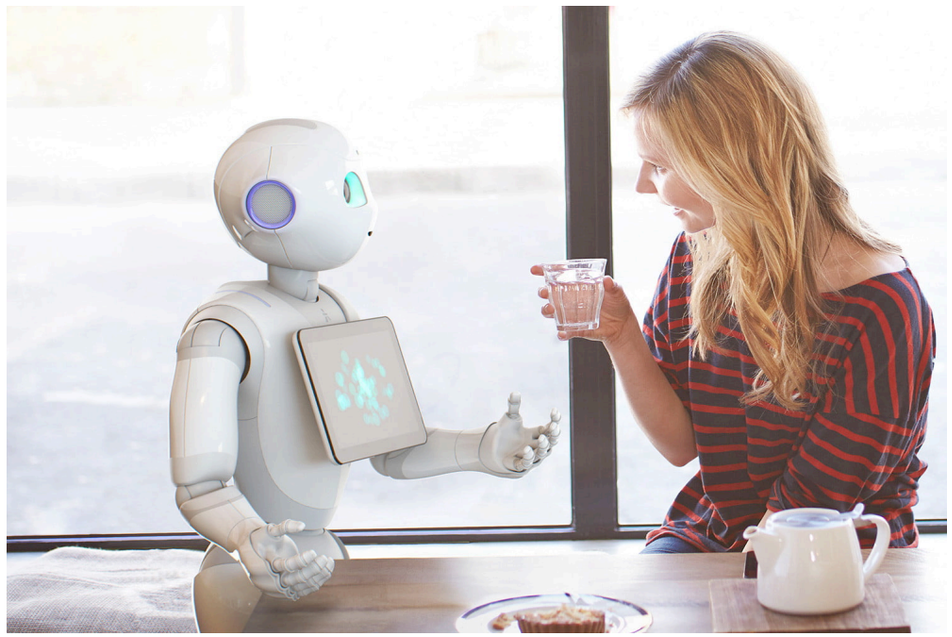
$$a_1 \rightarrow a_2$$

Lecture outline

- Recap stochastic local search
- Planning introduction
- Planning example 👉🏻
- STRIPS: a feature-based representation
- Forward planning
- Summary and wrap up

Example: Delivery robot (textbook)

Consider a delivery robot named Rob, who must navigate the following environment, can deliver coffee and mail to Sam



Delivery robot

Sam

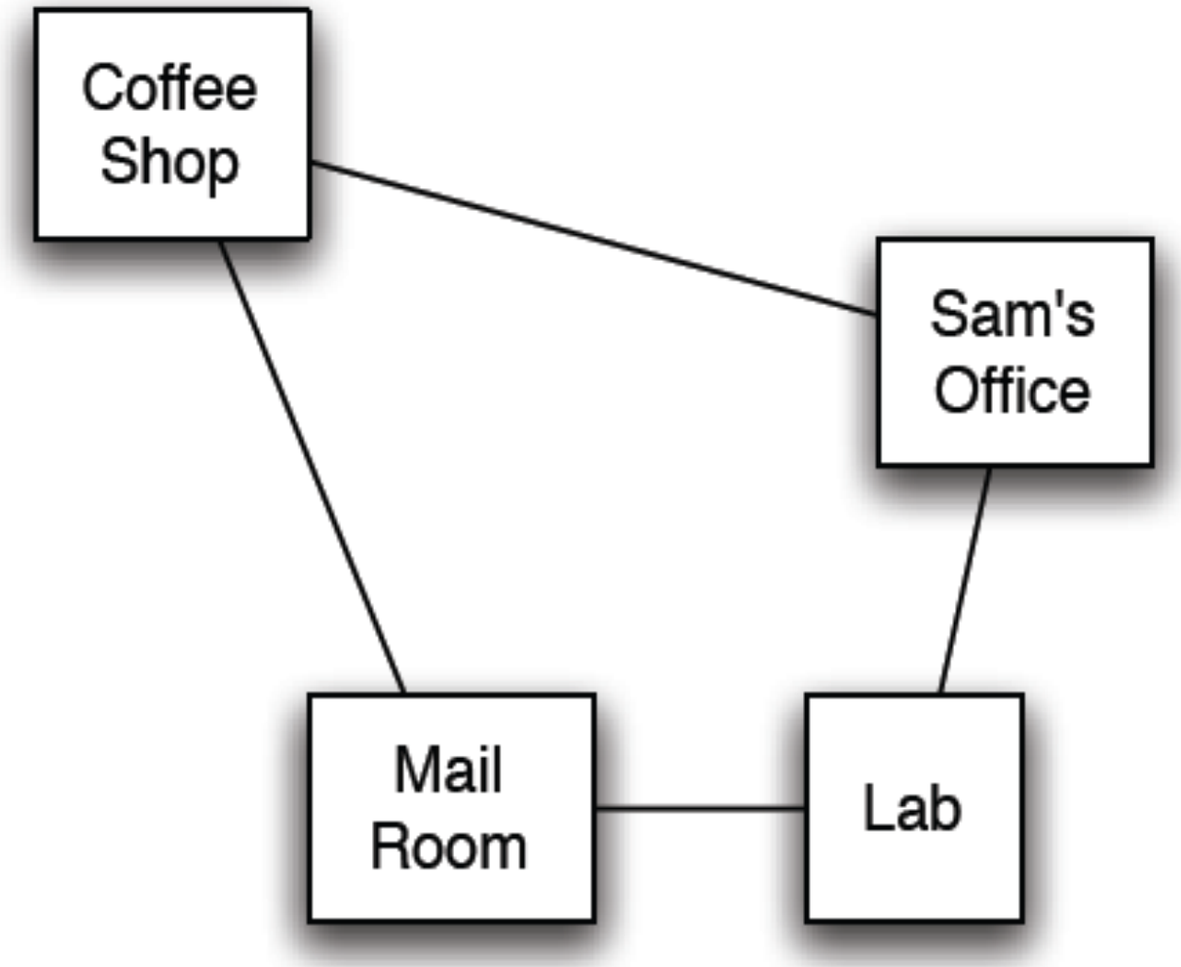
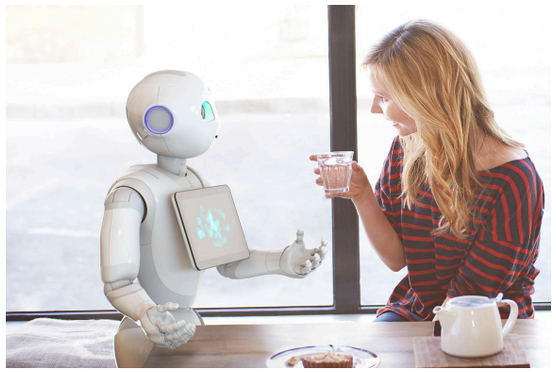


Image source: https://www.digitaltrends.com/cool-tech/things-machines-computers-still-cant-do-yet/

Example: Delivery robot



The state is defined by the following variables/features:

RLoc: Rob's location

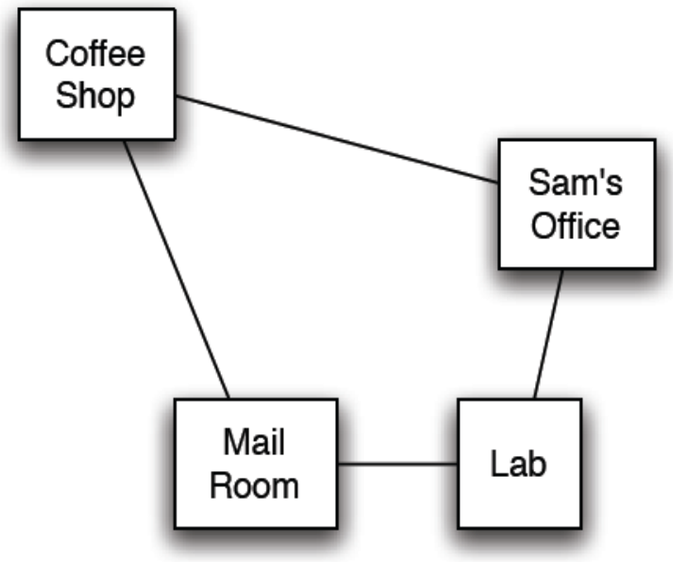
domain: {coffee shop (*cs*), Sam's office (*off*), mail room (*mr*), or laboratory (*lab*)}

RHC - Rob has coffee (T/F) $(rhc, \overline{rhc})$

SWC - Sam wants coffee (T/F) $(swc, \overline{swc})$

MW - Mail is waiting (T/F) $(mw, \overline{mw})$

RHM - Rob has mail (T/F) $(rhm, \overline{rhm})$

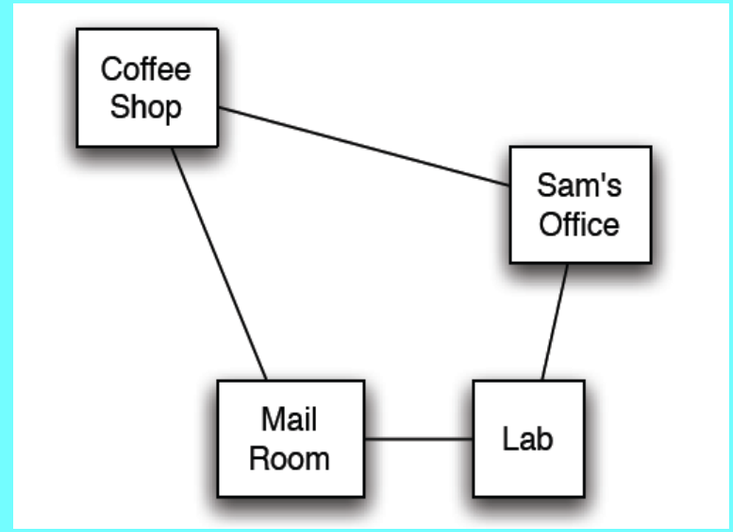


Example state:

$\{cs, rhc, \overline{swc}, \overline{mw}, rhm\}$

Example: Delivery robot

RLoc - Rob's location
domain: {coffee shop (*cs*), Sam's office (*off*), mail room (*mr*), or laboratory (*lab*)}
RHC - Rob has coffee (T/F) ($rhc, \overline{rhc}$)
SWC - Sam wants coffee (T/F) ($swc, \overline{swc}$)
MW - Mail is waiting (T/F) ($mw, \overline{mw}$)
RHM - Rob has mail (T/F) ($rhm, \overline{rhm}$)

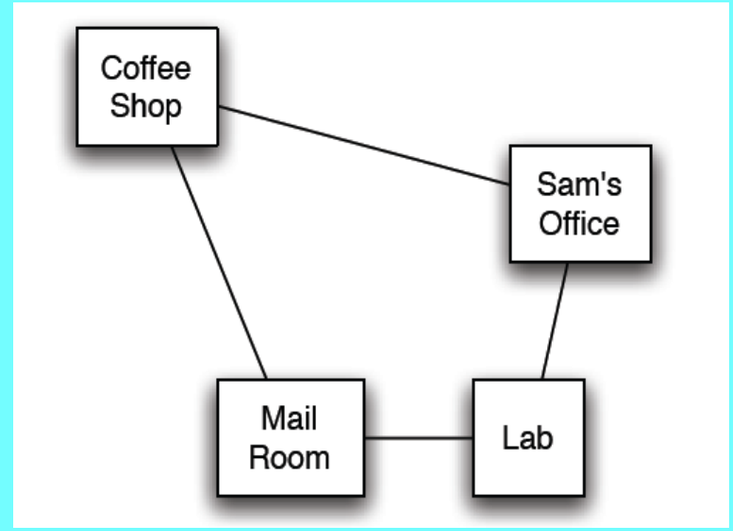


How many states?

A. 32 B. 64 C. 48 D. 16

Example: Delivery robot

RLoc - Rob's location
domain: {coffee shop (*cs*), Sam's office (*off*), mail room (*mr*), or laboratory (*lab*)}
RHC - Rob has coffee (T/F) ($rhc, \overline{rhc}$)
SWC - Sam wants coffee (T/F) ($swc, \overline{swc}$)
MW - Mail is waiting (T/F) ($mw, \overline{mw}$)
RHM - Rob has mail (T/F) ($rhm, \overline{rhm}$)

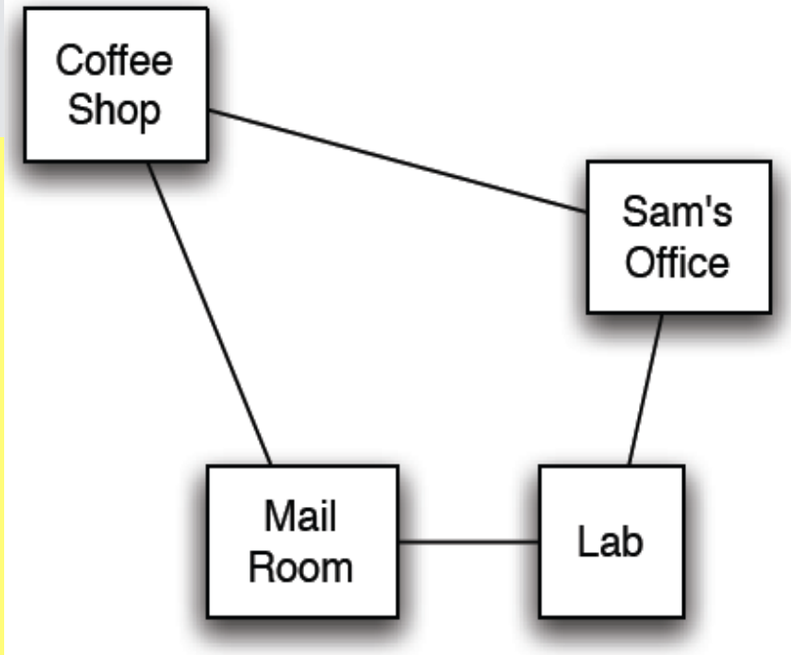


How many states?

A. 32 B. 64 ✅ C. 48 D. 16

Example: Delivery robot actions

Action	Preconditions
Move Rob's move action: Move clockwise (***mc***), Move counter-clockwise (***mcc***)	
***puc*:** Rob picks up coffee	Rob must be at the coffee shop
***dc*:** Rob delivers coffee	Rob must be at the office and must have coffee
***pum*:** Rob picks up mail	Rob must be in the mail room, and mail must be waiting
dm: Rob delivers mail	Rob must be at the office and have mail



Example: Delivery robot actions

Action	Preconditions	Effects/Results
Move Rob's move action: Move clockwise (***mc***), Move counter-clockwise (***mcc***)		
***puc*:** Rob picks up coffee	Rob must be at the coffee shop	Rob has coffee
***dc*:** Rob delivers coffee	Rob must be at the office and must have coffee	Rob does not have coffee; Sam does not want coffee
***pum*:** Rob picks up mail	Rob must be in the mail room, and mail must be waiting	Rob has mail, mail is not waiting
dm: Rob delivers mail	Rob must be at the office and have mail	Rob does not have mail

Example: Explicit state-space representation

Tabular representation: Need an entry for every state and every action applicable in that state.

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

Need more compact representation.

Example of a more compact representation

A representation of the action pick up coffee, ***puc***:

- Only changes **a subset of features**. In this case, only RHC (Rob has coffee).

- Only depends on **a subset of features**. In this case, $RLoc = cs$ (Rob is in the coffee shop)

- **Preconditions** $RLoc = cs$ and $RHC = \overline{rhc}$

- **Effects** $RHC = rhc$

Lecture outline

- Recap stochastic local search
- Planning introduction
- Planning example
- STRIPS: a feature-based representation 👉
- Forward planning
- Summary and wrap up

STRIPS action representation

The key to sophisticated planning is modelling actions.

STRIPS: STanford Research Institute Problem Solver

- The planner in Shakey, first AI robot

In STRIPS, an action has two parts:

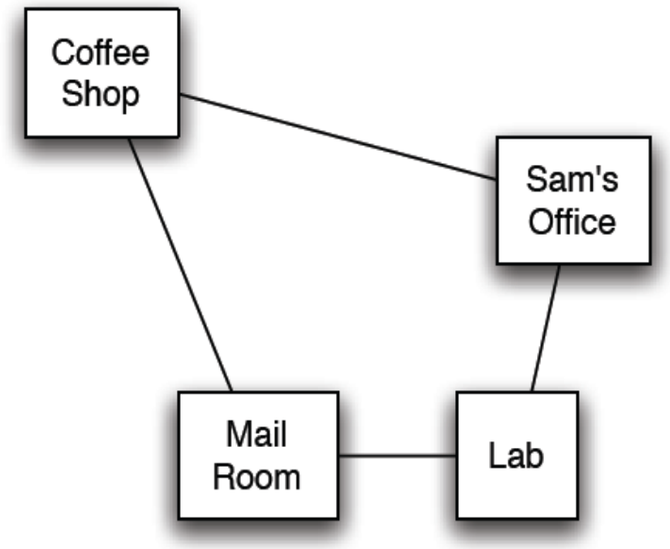
1. **Preconditions**: a set of assignments to features that **must be satisfied** in order for the action to be legal

2. **Effects**: a set of assignments to features that are **caused** by the action

STRIPS actions: Example (pair-share)

STRIPS representation of the action pick up coffee ***puc***:
preconditions: $RLoc = cs$ and $RHC = \overline{rhc}$
effects: $RHC = rhc$

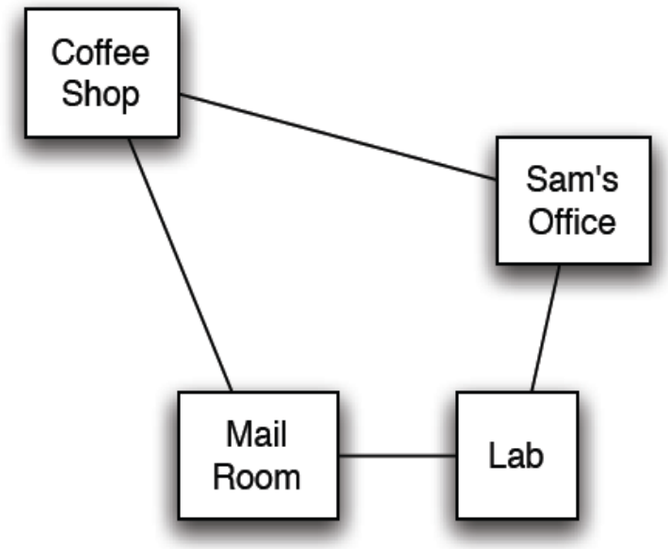
STRIPS representation of the action deliver coffee, ***dc***:
preconditions: $RLoc = ?, RHC = ?$
effects: $RHC = ?, SWC = ?$



STRIPS actions: Example

STRIPS representation of the action pick up coffee ***puc***:
preconditions: $RLoc = cs$ and $RHC = \overline{rhc}$
effects: $RHC = rhc$

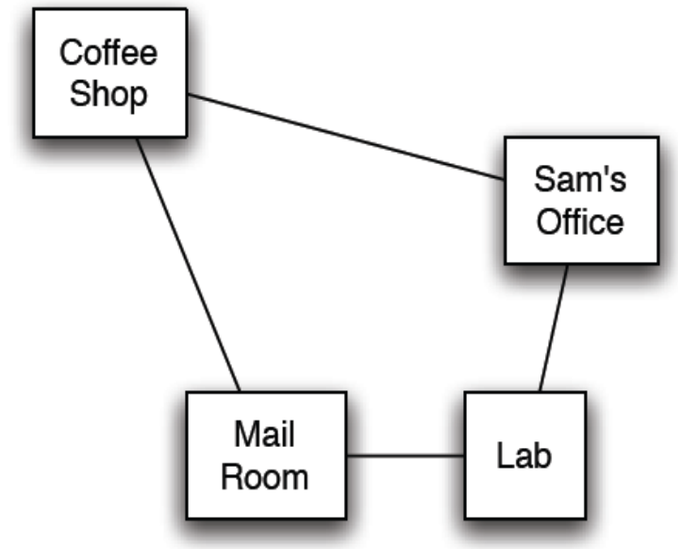
STRIPS representation of the action deliver coffee, ***dc***:
preconditions: $RLoc = off, RHC = rhc$
effects: $RHC = \overline{rhc}, SWC = \overline{swc}$



STRIPS actions: Example

STRIPS representation of the action pick up coffee ***puc***:
preconditions: $RLoc = cs$ and $RHC = \overline{rhc}$
effects: $RHC = rhc$

STRIPS representation of the action deliver coffee, ***dc***:
preconditions: $RLoc = off, RHC = rhc$
effects: $RHC = \overline{rhc}, SWC = \overline{swc}$



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.

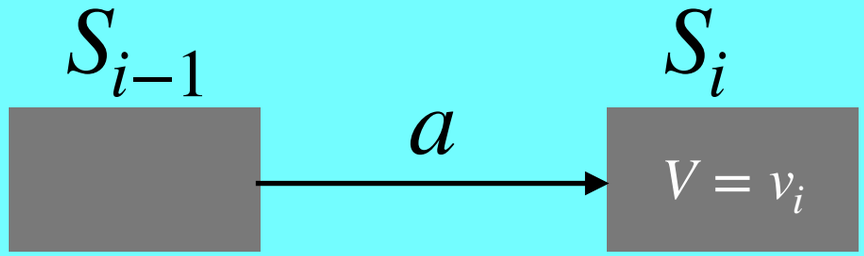
The STRIPS assumption

All features not explicitly changed by an action stay unchanged

STRIPS actions

All features not explicitly changed by an action stay unchanged.

What can we conclude about action a and/or the state of the world S_{i-1} immediately preceding the execution of a?



A. V was also v_i in S_{i-1}

B. One of the effects of a is to set $V = v_i$

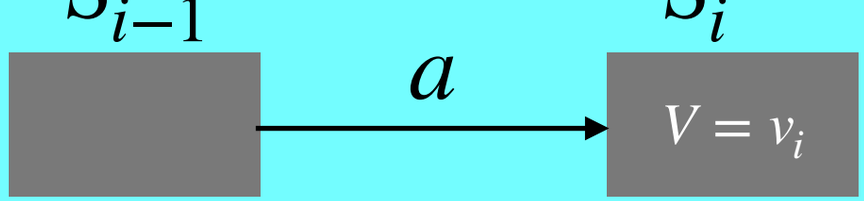
C. At least one of the above

D. None of the above

STRIPS actions

All features not explicitly changed by an action stay unchanged.

What can we conclude about action a and/or the state of the world S_{i-1} immediately preceding the execution of a?



A. V was also v_i in S_{i-1}

B. One of the effects of a is to set $V = v_i$

C. At least one of the above ✅

D. None of the above

Lecture outline

- Recap stochastic local search
- Planning introduction
- Planning example
- STRIPS: a feature-based representation
- Forward planning 👉
- Summary and wrap up

Forward planning

Idea: To find a plan (i.e., a solution), search in the state-space graph using any of the search strategies we have learned so far

- The **states** are the **possible worlds**
- 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

Forward planning

What actions a are legal/possible in state s?

A. Those where a's effects are satisfied in s

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

C. Those where that state s' reached via a is on the way to the goal

D. Those where a's preconditions and effects are the same

Forward planning

What actions a are legal/possible in state s?

A. Those where a's effects are satisfied in s

B. Those where a's preconditions are satisfied in s ✅

C. Those where that state s' reached via a is on the way to the goal

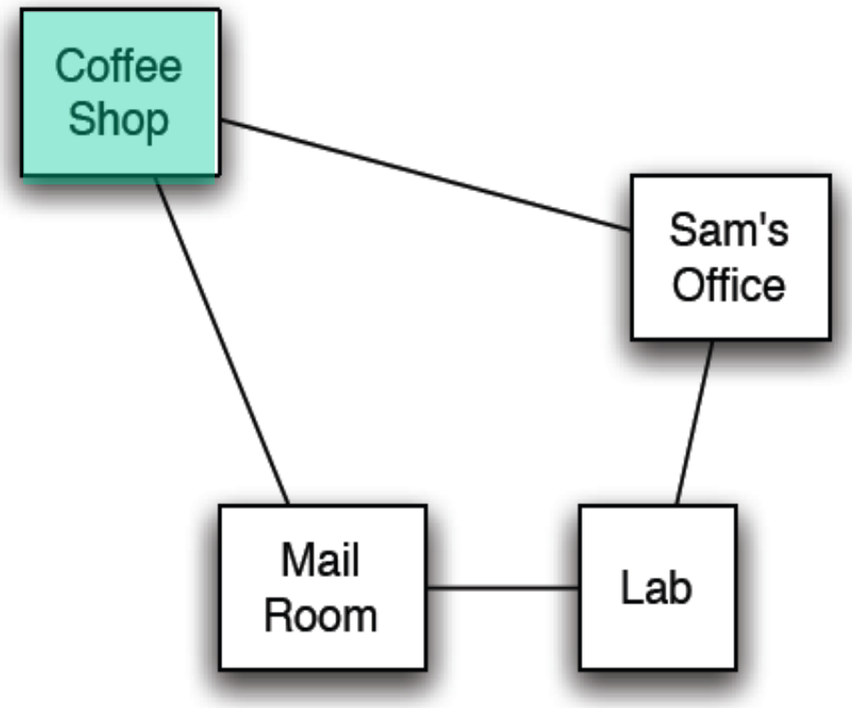
D. Those where a's preconditions and effects are the same

Example: state-space planner (first level)

Actions

mc: move clockwise
mcc: move counter-clockwise

puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail



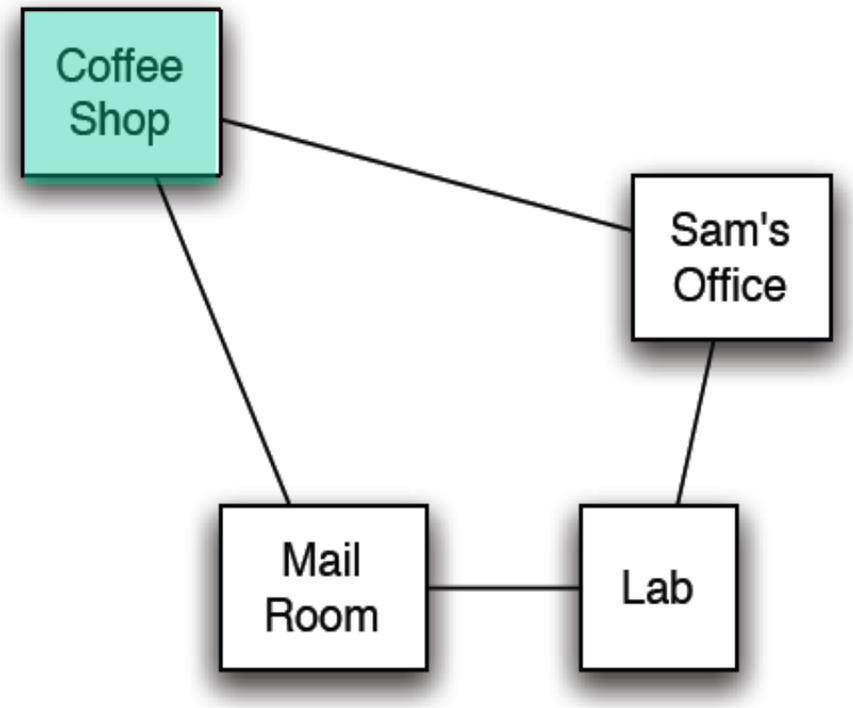
⟨*cs*, ¬*rhc*, *swc*, *mw*, ¬*rhm*⟩

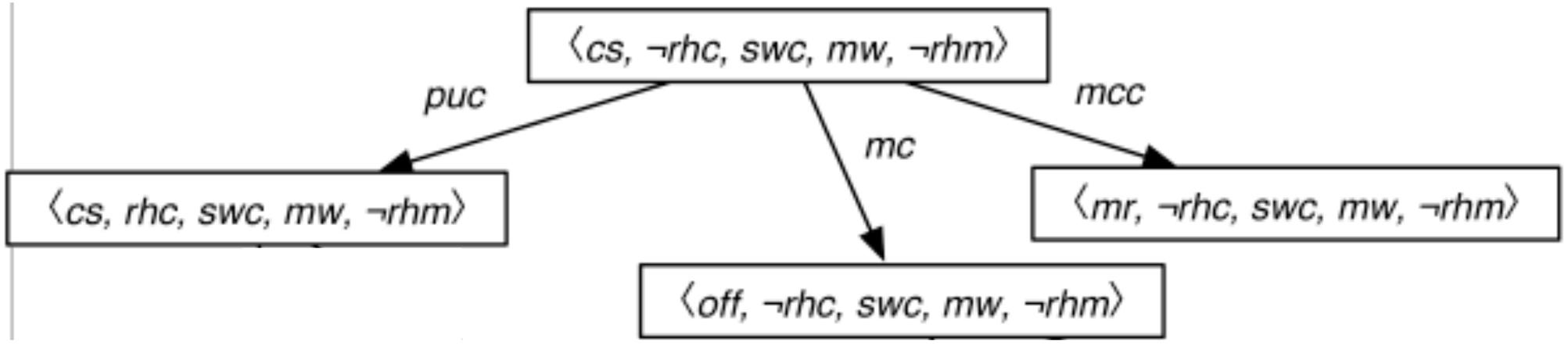
Example: state-space planner (first level)

Actions

mc: move clockwise
mcc: move counter-clockwise

puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail

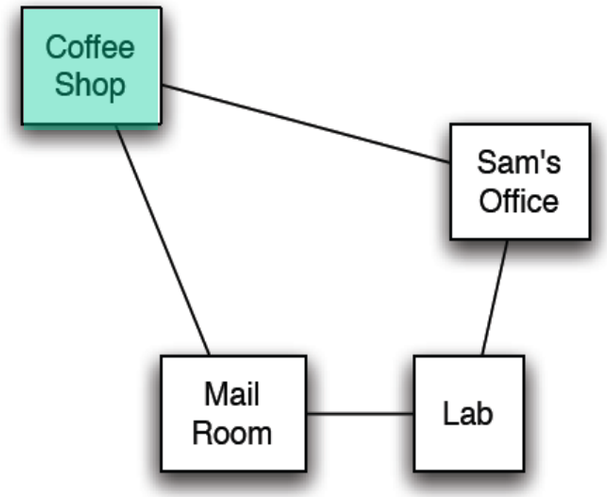


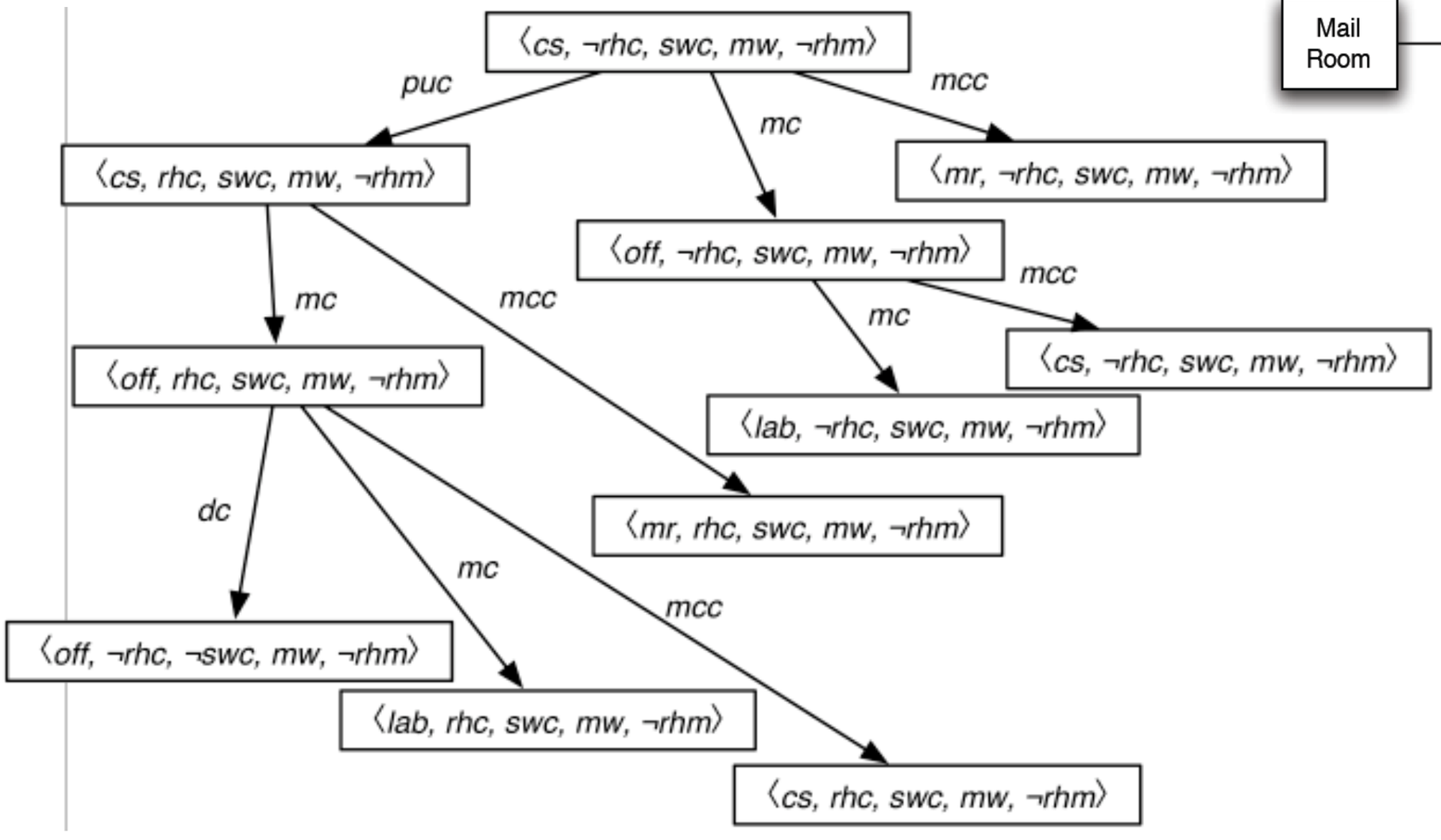


Example: part of state-space planner

Goal: $\overline{swc}$

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





Example: solution in state-space graph

Goal: $\overline{swc}$ ($\neg swc$)
Solution: A sequence of actions that gets us from the start to the goal

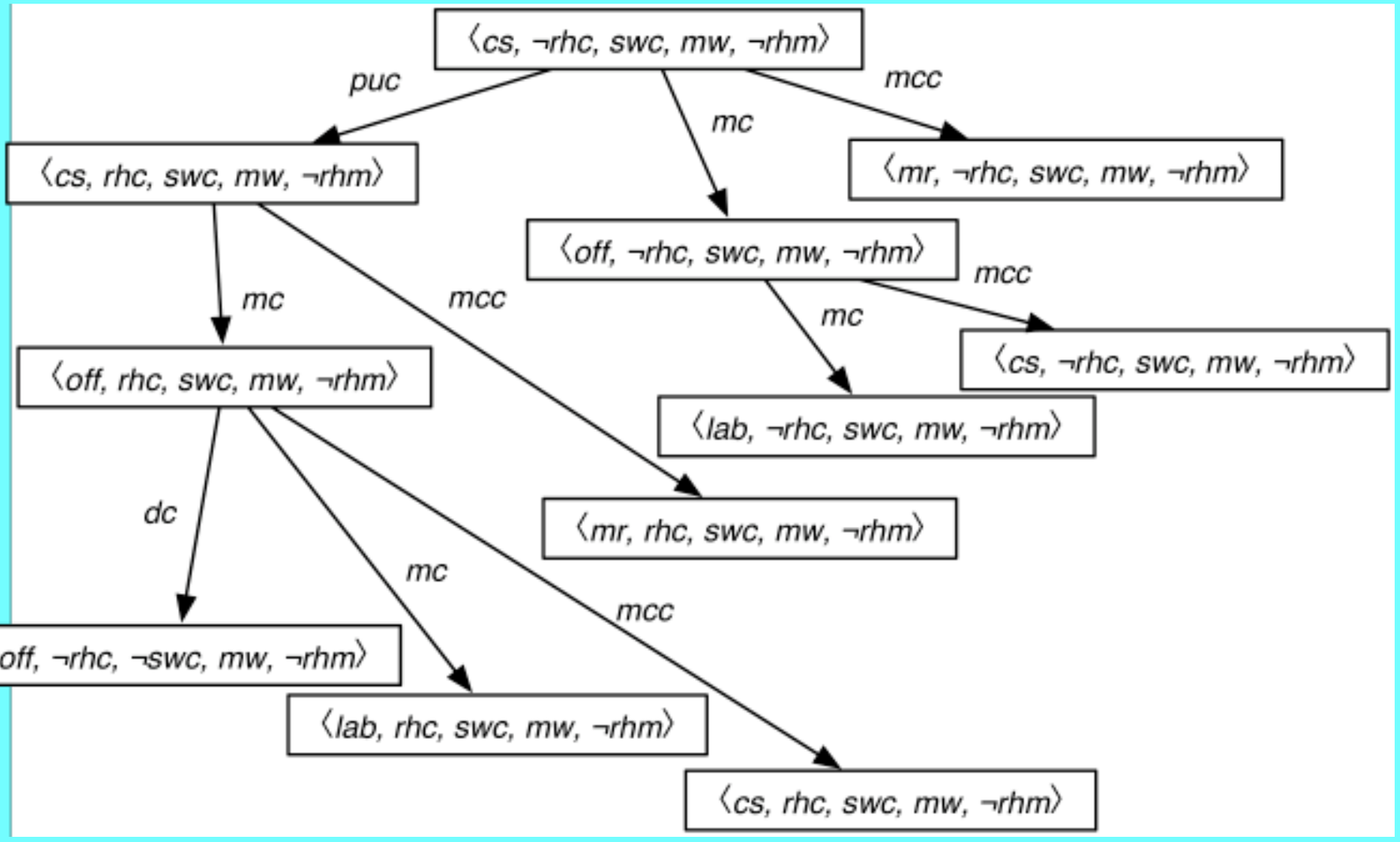
What's the solution to this planning problem?

A. *(puc, mc)*

B. *(puc, mc, mc)*

C. *(puc, dc)*

D. *(puc, mc, dc)*



Example: solution in state-space graph

Goal: $\overline{swc}$ ($\neg swc$)
Solution: A sequence of actions that gets us from the start to the goal

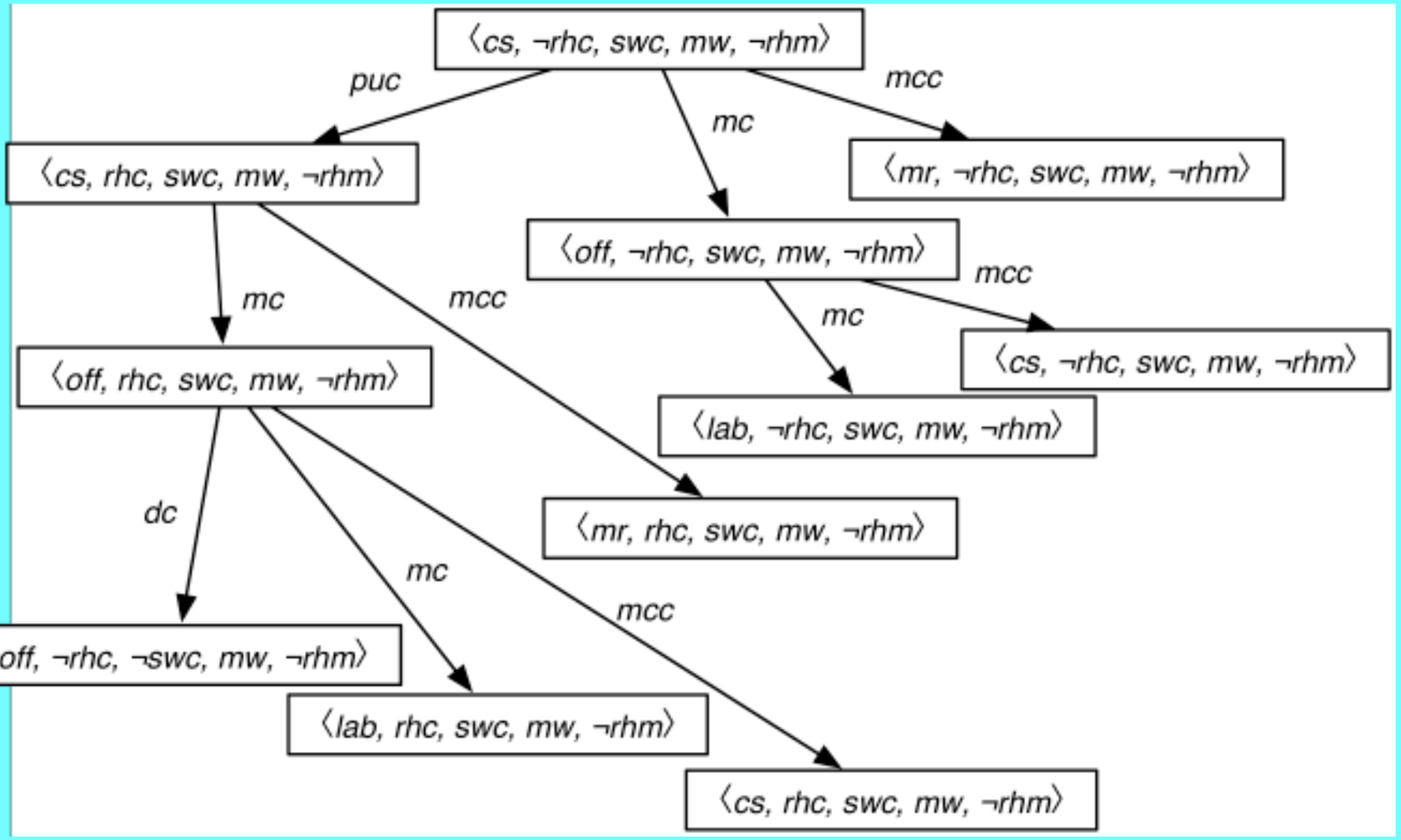
What's the solution to this planning problem?

A. *(puc, mc)*

B. *(puc, mc, mc)*

C. *(puc, dc)*

D. *(puc, mc, dc)* ✅



Forward planning vs. Tabular representation

Why is using a forward planner not the same as making an explicit state-based (tabular representation)?

Forward planning vs. Tabular representation

Why is using a forward planner not the same as making an explicit state-based (tabular representation)?

the relevant part of the graph is created dynamically from the representations of the actions.

Planning: Summary

State: full assignment of values to variables

Successor function: actions with preconditions/effects

Goal test: partial (or full) assignment of values to variables

Solution: sequence of actions to goal

Heuristic function: (next class)

Revisit: Learning outcomes

From this lecture, students are expected to be able to:

- Represent a planning problem with the STRIPS representation

- Explain the STRIPS assumption

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

Coming up

Planning: How to select and organize a sequence of actions to achieve a given goal…

Readings for next class

- 6.2 Forward Planning
- 6.4 Planning as a CSP