CPSC 322: Introduction to Artificial Intelligence

Planning: Heuristics and CSP Planning

Planning

Textbook reference: [6.4]

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

- Assignment 3 has been released.
  - Due date: **Nov 11, 11:59 PM**
- Final exam scheduled: **Dec 9 at 7:00pm**

- We have started grading the midterm and we should be able to return it by the end of this week if everything goes well.

Lecture outline

- Recap: Planning representation and STRIPS 👉🏻
- Heuristics for forward planning
- CSP planning

Why planning?

- Intelligent agents must operate in the world.

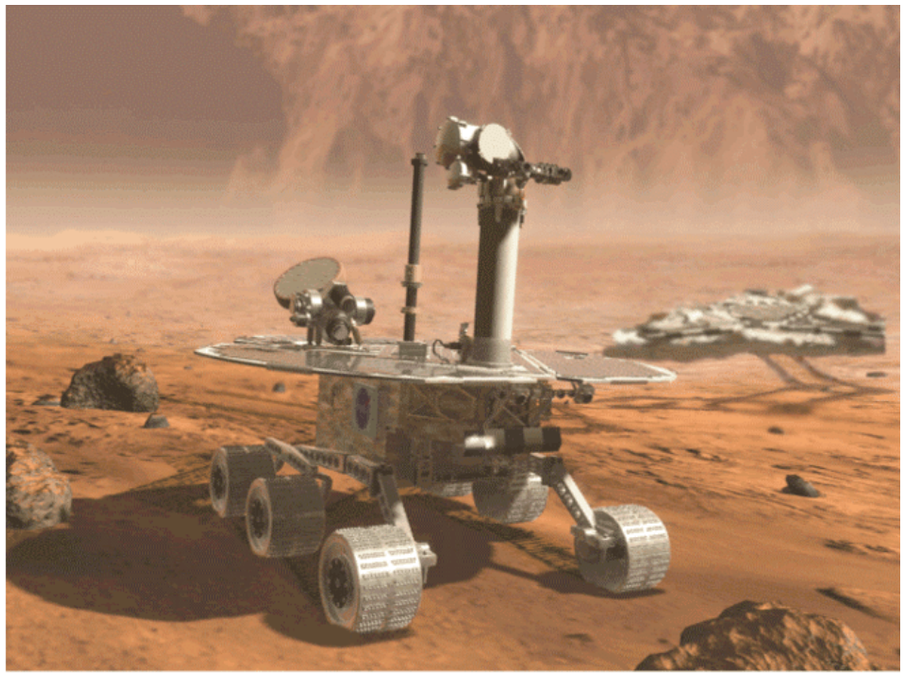
- They are not simply passive reasoners (Knowledge Representation, reasoning under uncertainty) or problem solvers (Search), they must also act in the world

- 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

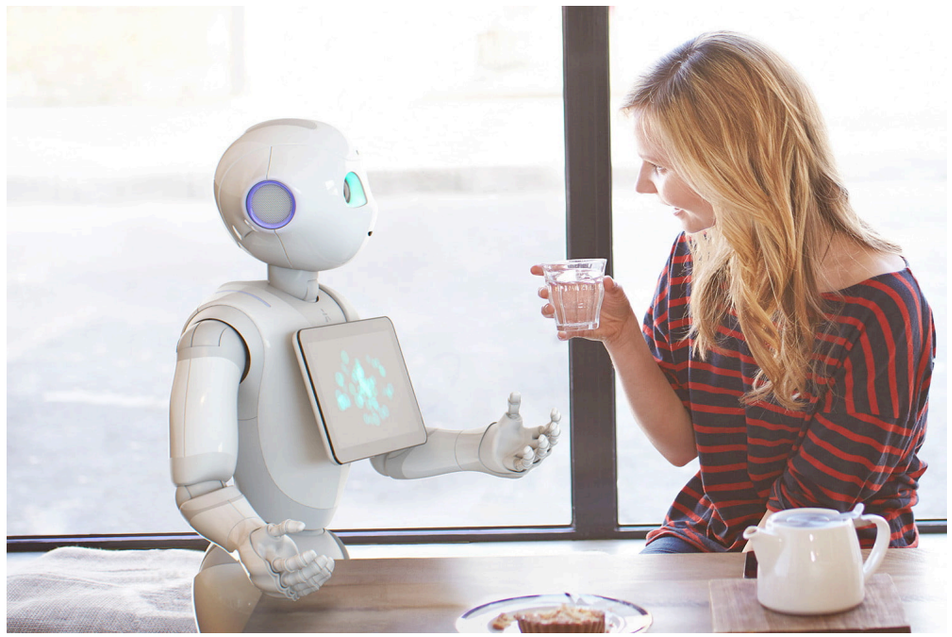
Mars exploration rover



Coffee delivery drone

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

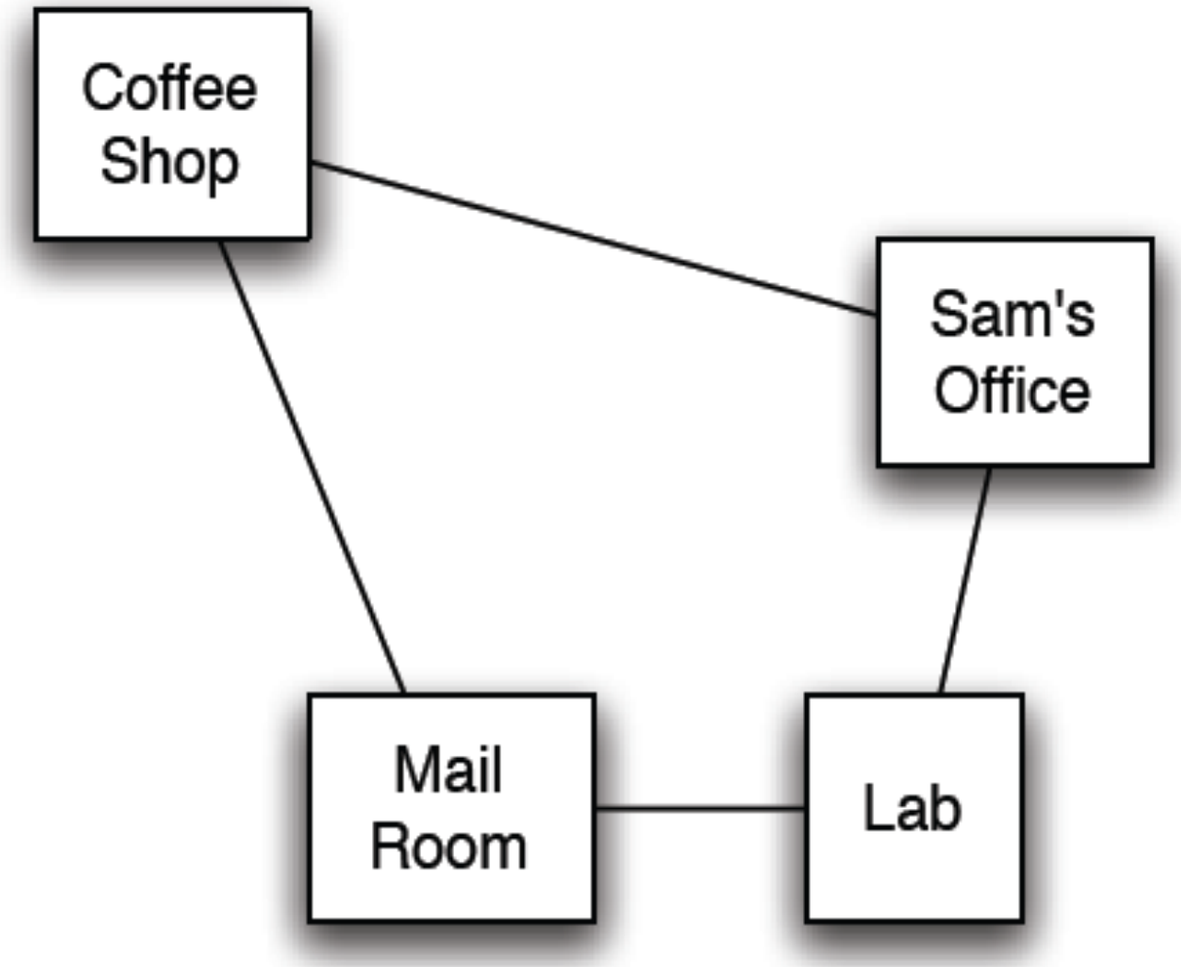
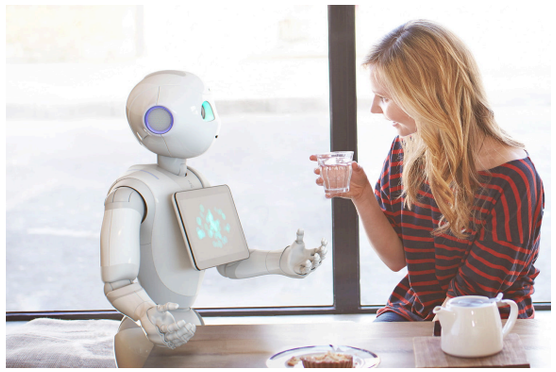


Image source: https://www.digitaltrends.com/cool-tech/things-machines-computers-still-cant-do-yet/

Example: Delivery robot



The state is defined by the following variables/features:

RLoc: Rob's location

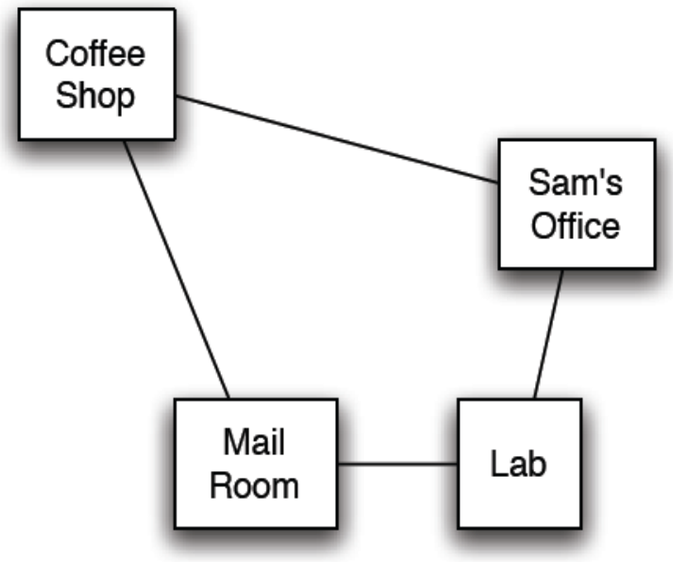
domain: {coffee shop (*cs*), Sam's office (*off*), mail room (*mr*), or laboratory (*lab*)}

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:

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

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

Example: Delivery robot actions

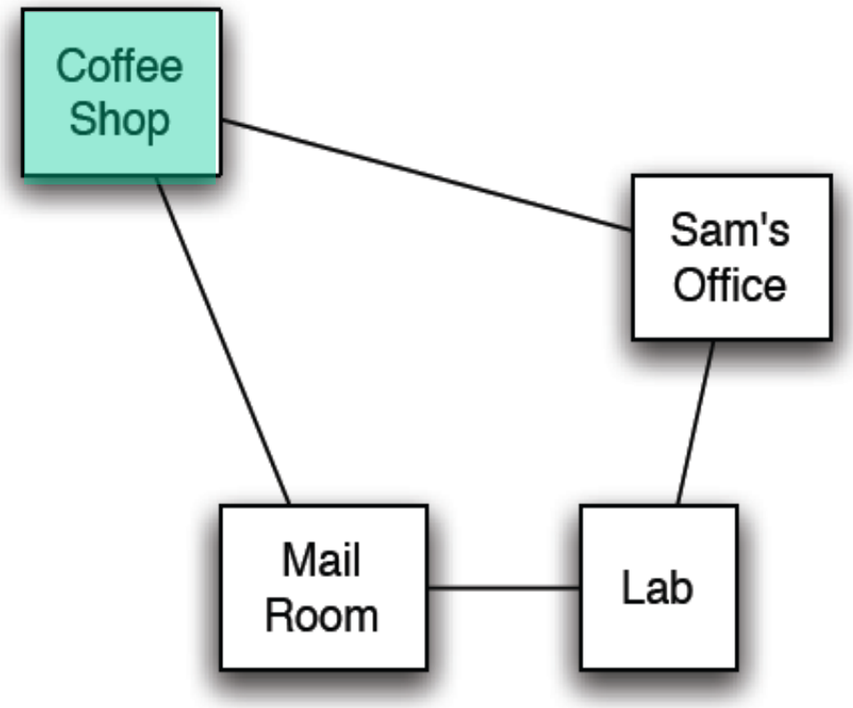
Action	Preconditions	Effects/Results
Move Rob's move action: Move clockwise (***mc***), Move counter-clockwise (***mcc***)		
***puc*:** Rob picks up coffee	Rob must be at the coffee shop	Rob has coffee
***dc*:** Rob delivers coffee	Rob must be at the office and must have coffee	Rob does not have coffee;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

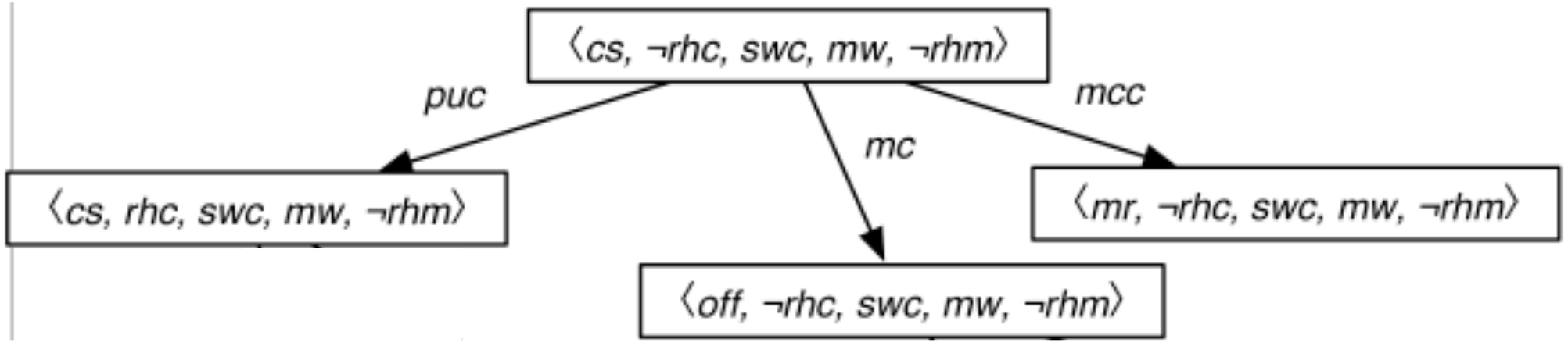
Example: state-space planner (first level)

Actions

mc: move clockwise
mcc: move counter-clockwise

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

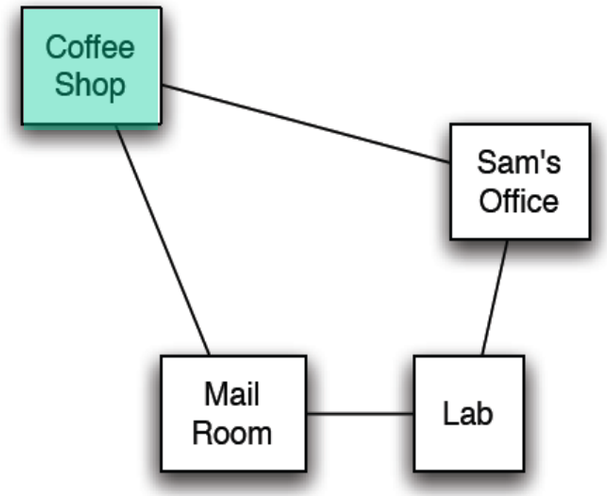


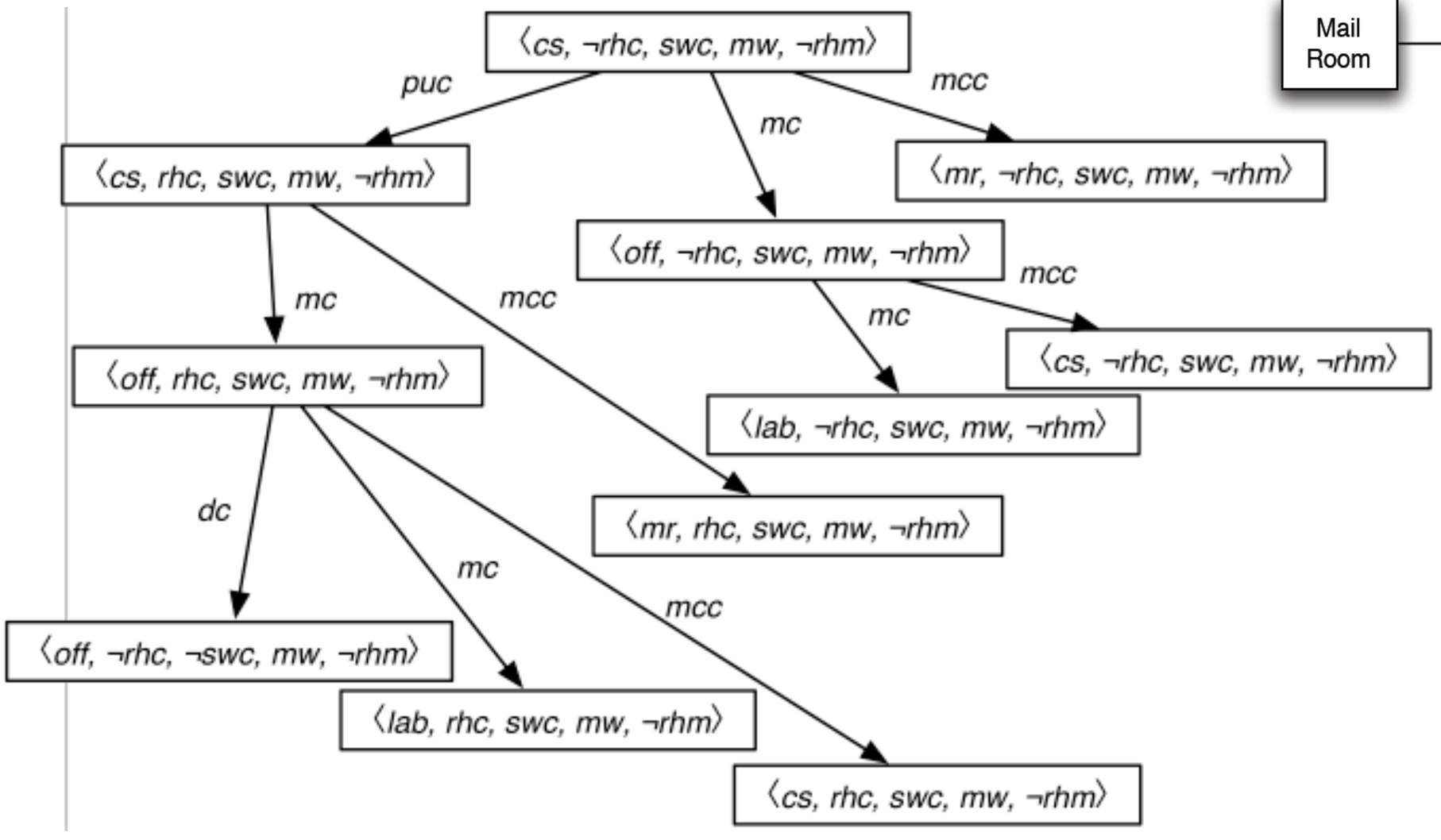


Example: Forward planning (part of state-space planner)

Goal: $\overline{swc}$

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



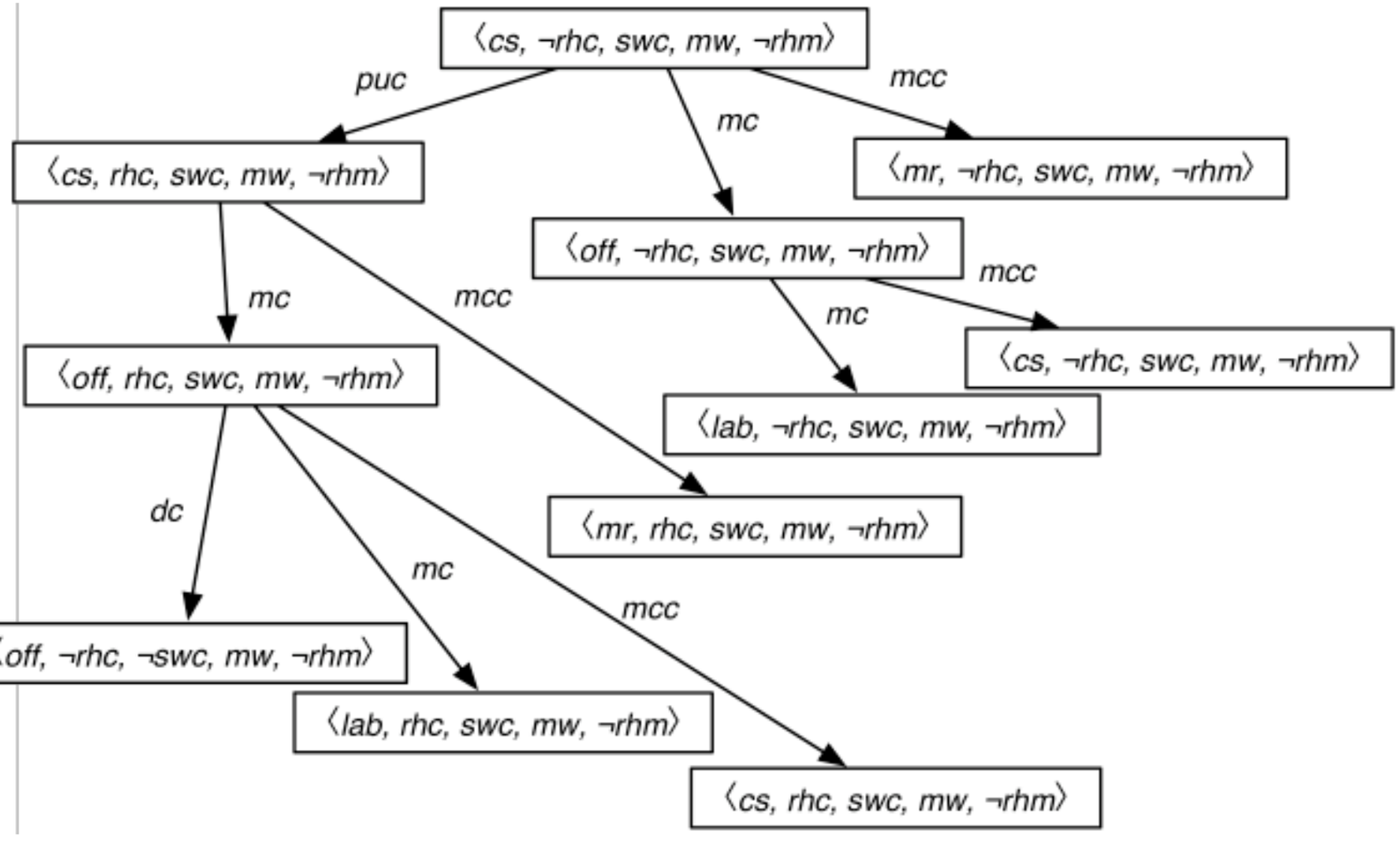


Example: solution in state-space graph

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?

(puc, mc, dc)



Planning (pair-share)

State

A. Full assignment ✅

B. Partial assignment

Successor function

A. Actions ✅

B. Random assignments

Goal test

A. Partial assignment ✅

B. Full assignment

C. It depends

Solution

A. A goal state

✅

B. A path to a goal state

Today: Learning outcomes

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

- Construct and justify a heuristic function for forward planning.

- Translate a planning problem represented in STRIPS into a corresponding CSP problem

- Solve a planning problem with CSP by expanding the horizon

Lecture outline

- Recap: Planning representation and STRIPS
- Heuristics for forward planning 👉
- CSP planning

Heuristics for forward planning

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

Heuristic for forward planning

i-clicker

Heuristic function:
an **estimate** of the **distance** from a state to the goal

In planning, the distance from a state s to the goal is

A. goal features not true in state s

B. actions needed to get from s to the goal ✅

C. legal actions in s

Heuristic for forward planning

In planning this is **the number of actions**. Good heuristics makes forward planning feasible in practice.

- Recall the general method to create heuristics: Relax the original problem

- Factored representation of states and actions allows for definition of domain-independent heuristics.

- We will see two examples of two general, domain-independent heuristics.

Forward planning: Admissible heuristics

For simplicity assume that

- All features are binary: T / F
- Goals and preconditions can only be assignments to T

Forward planning: Heuristics

Let's stay in the coffee delivery robot domain with a small change:

Let's say our robot has to bring coffee to **Bob**, **Sue**, and **Steve**:

G = {bob_has_coffee = T, sue_has_coffee = T, steve_has_coffee = T}

They all sit in different offices.

Forward planning: Admissible heuristics I

Ignore preconditions heuristics

- Heuristic drops all preconditions from actions.
- Any action can be applied to any state

Count how many subgoals are not achieved yet

Example:
Subgoal_1: bob_has_coffee = T
Subgoal_2: sue_has_coffee = T
Subgoal_3: steve_has_coffee = T

Simply apply "DeliverCoffee(person)" action for each person

Admissible but not very informative.

Forward planning: Admissible heuristics 2

Ignore "delete lists" heuristics

- Every action has **add list** and **delete list**
  - Add list: features that are made true by the action
    Example: Add list for "pick-up coffee": *rhc*
  - Delete list: features that are made false by the action
    Example: Delete list for "deliver coffee": *rhc*
    "Ignore delete lists" ⇔ once you have coffee you keep it

Forward planning: Admissible heuristics 2

Ignore "delete lists" heuristics

- Rewrite effects as add and delete lists and **ignore** delete lists.

- Problem gets easier
  Example: only need to pick up coffee once, navigate to the right locations, and deliver

- Admissible and typically more realistic than ignoring preconditions

This heuristics with forward planning is currently considered a very successful strategy

Ignore "delete lists" heuristics: Example

Example:
Variables: A, B, C
Domains: {True (T), False (F)}

Start state

$$\begin{bmatrix} A = T \\ B = F \\ C = F \end{bmatrix} \ldots$$

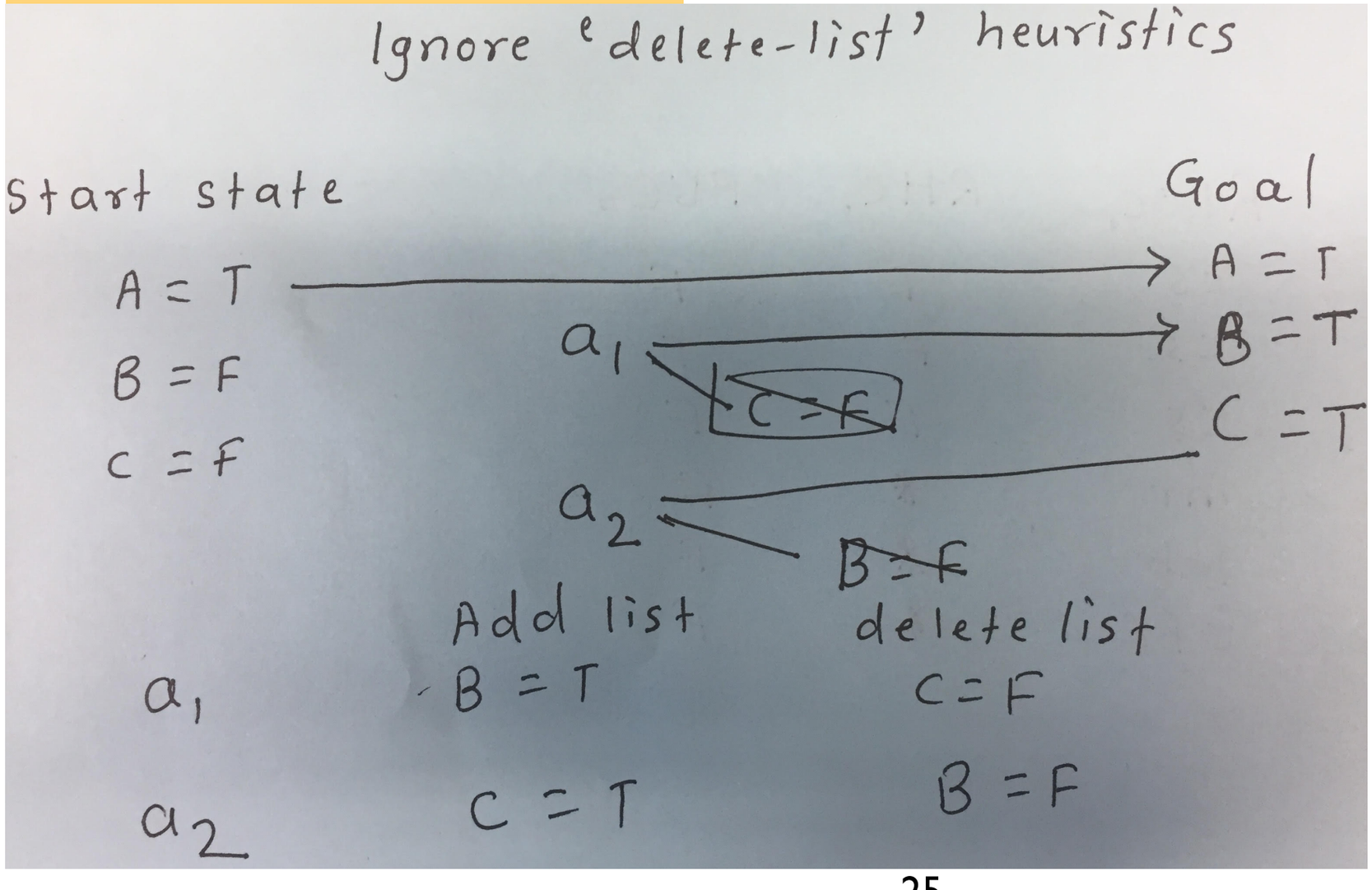
Goal

$$\begin{bmatrix} A = T \\ B = T \\ C = T \end{bmatrix}$$

Ignore "delete lists" heuristics: Example

Example:
Variables: A, B, C
Domains: {True (T), False (F)}

Start state: $\begin{bmatrix} A = T \\ B = F \\ C = F \end{bmatrix}$, Goal: $\begin{bmatrix} A = T \\ B = T \\ C = T \end{bmatrix}$



Ignore delete list heuristics in practice

To compute $h(s_i)$

- Run forward planner with s_i as the start state, with the same goal as the original problem but with all the negative effects of actions removed.

- So to compute h, we need to solve a planning problem but a simpler one.

Final comment on heuristics for forward planning

- You should view (informed) forward planning as one of the basic planning techniques.

- By itself, it cannot go far, but it can work very well in combination with other techniques, for specific domains.

Lecture outline

- Recap: Planning representation and STRIPS
- Heuristics for forward planning
- CSP planning 👉

Planning as CSP

- An alternative approach to planning is to set up a planning problem as a CSP!

- We simply reformulate a STRIPS model as a set of variables and constraints

- Once this is done we can even express additional aspects of our problem (as additional constraints)

Planning as CSP (pair-share)

Idea: reformulate a STRIPS model as a set of variables and constraints.

What would be variables in this CSP?

A. The STRIPS variables

B. The values of the STRIPS variables

C. The STRIPS actions

D. The STRIPS preconditions

Planning as CSP overview

We need to "unroll the plan" for a fixed number of steps: this is called the **horizon**.

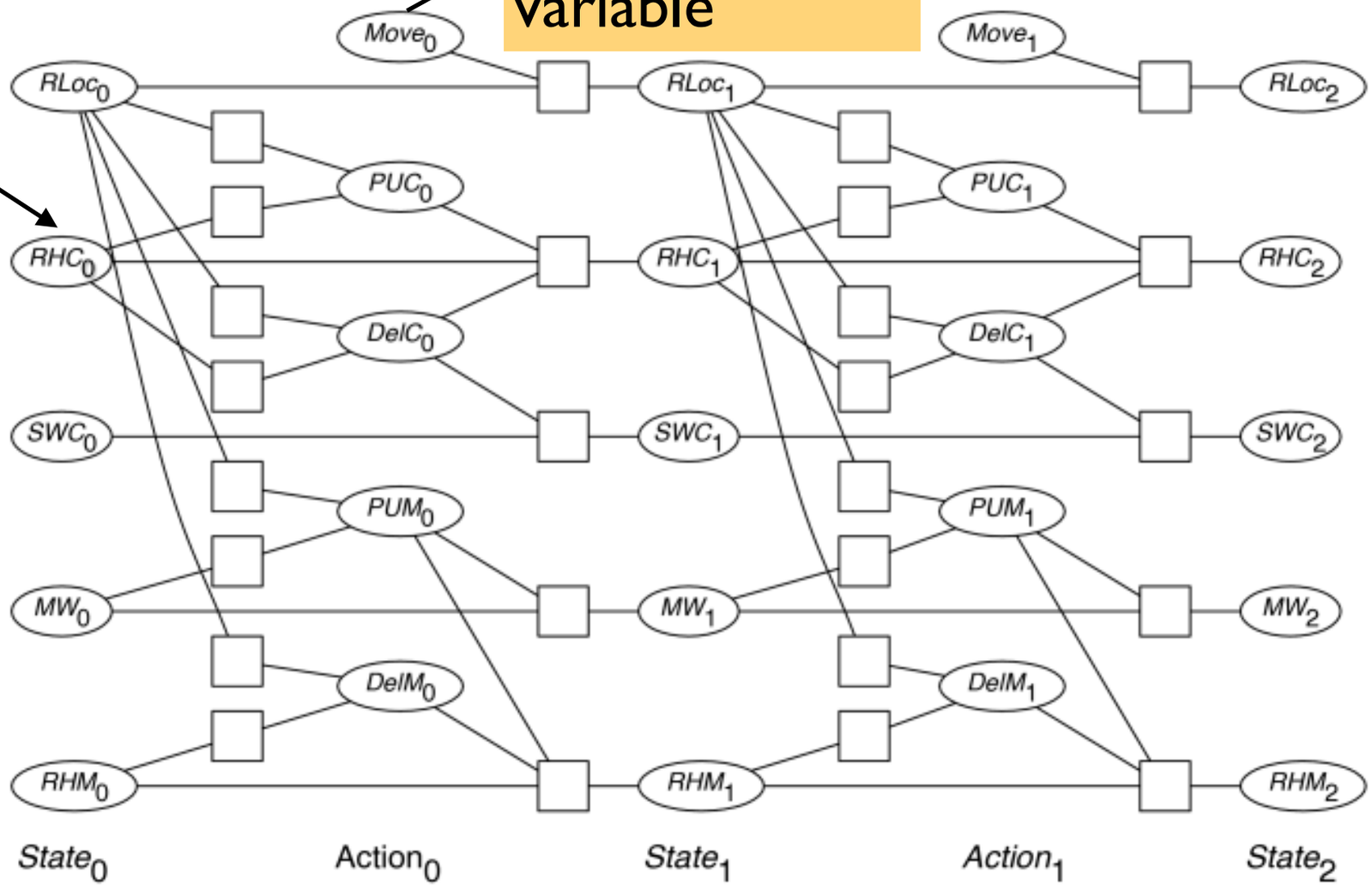
To do this with a horizon of k:

- Construct a CSP **variable for each STRIPS variable** (e.g., *RLoc, RHS, SWC*) at each time step from 0 to k
- Construct a boolean CSP **variable for each STRIPS action** (e.g., move, puc, dc) at each time step from 0 to $k-1$.
- Construct **CSP constraints** corresponding to start and goal values, as well as preconditions and effects of actions

CSP planning: robot example

STRIPS state variable

STRIPS action variable



Variables for actions are binary:
whether action occurs or not at that step

Planning as a CSP: main constraints

- Between actions at time t and previous state variables (time t)
  - When does an action apply? (precondition constraints)
- Between actions at time t and following state variables (time t)
  - How does an action change the variables? (effect constraints)

CSP planning: initial state and goal constraints

Unary

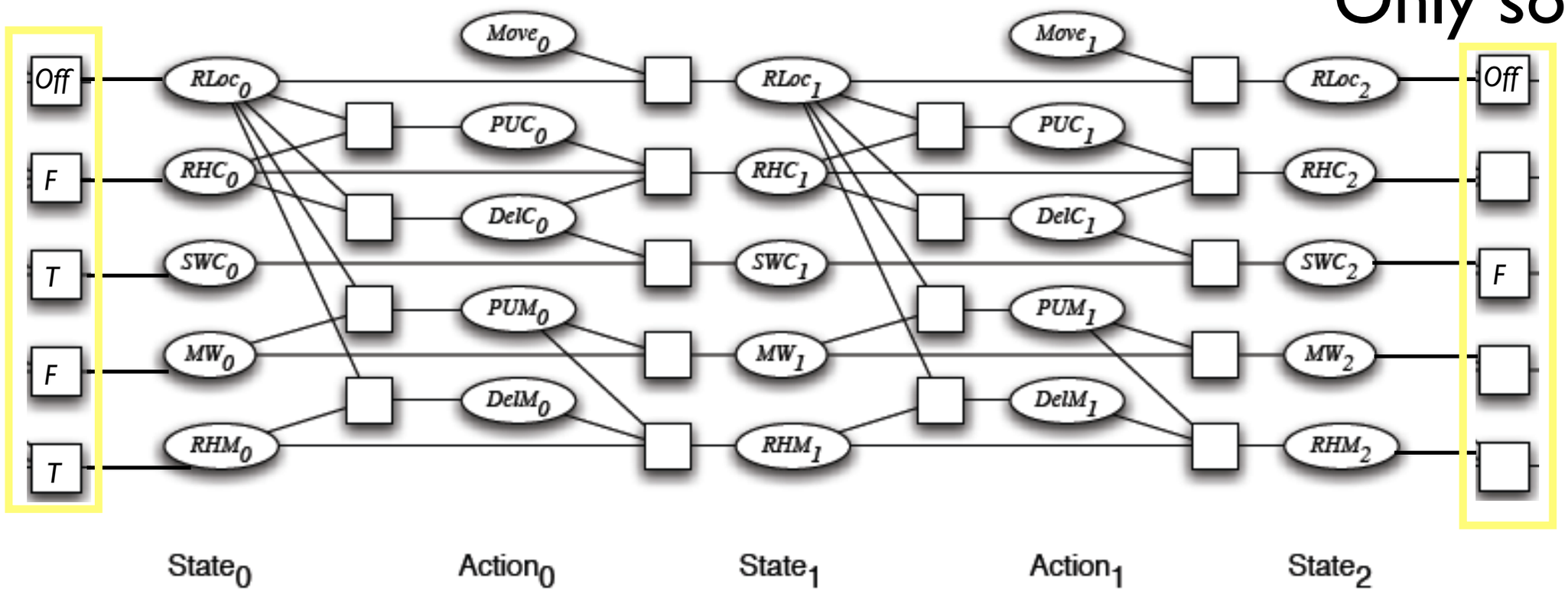
Initial state constraints constrain the state variables at time 0

Example: Start condition can be Sam wants coffee

Goal constraints constrain the state variables at time k.

Example: Goal condition can be Sam doesn't want coffee

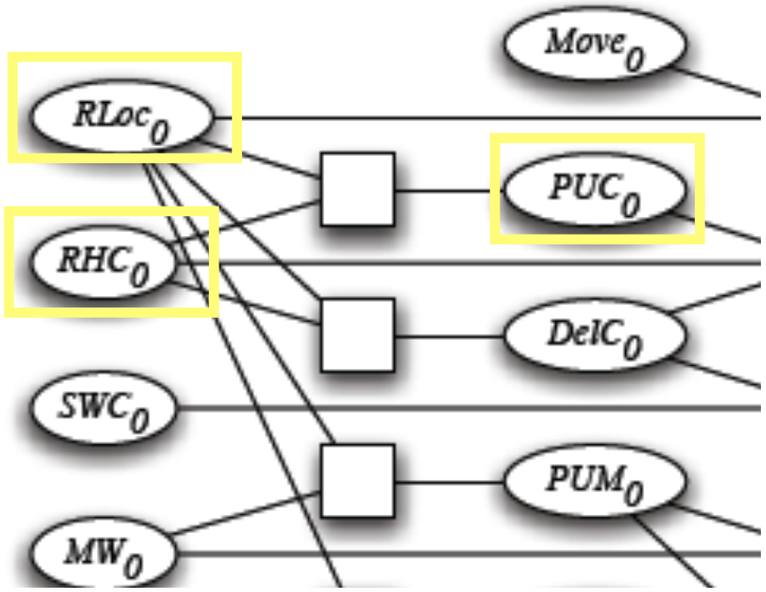
Only some



CSP planning: Precondition constraints

- Hold 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 *RLoc*=cs and RHC = false (don't have coffee already)

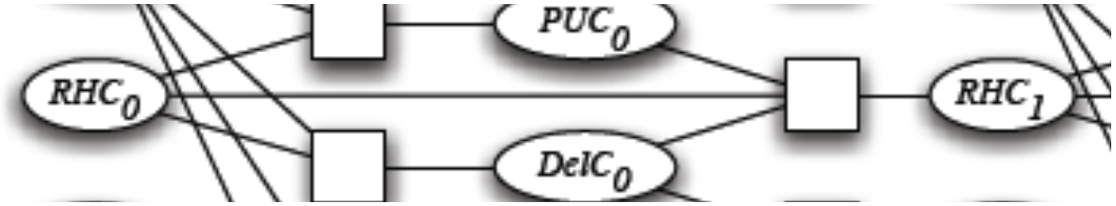


PUC_0

$RLoc_0$	RHC_0	PUC_0
cs	T	F
cs	F	T
cs	F	F
mr	*	F
lab	*	F
off	*	F

CSP planning: Effects constraints

Effect constraints



- between state variables at time t, action variables at time t and state variables at time $t+1$

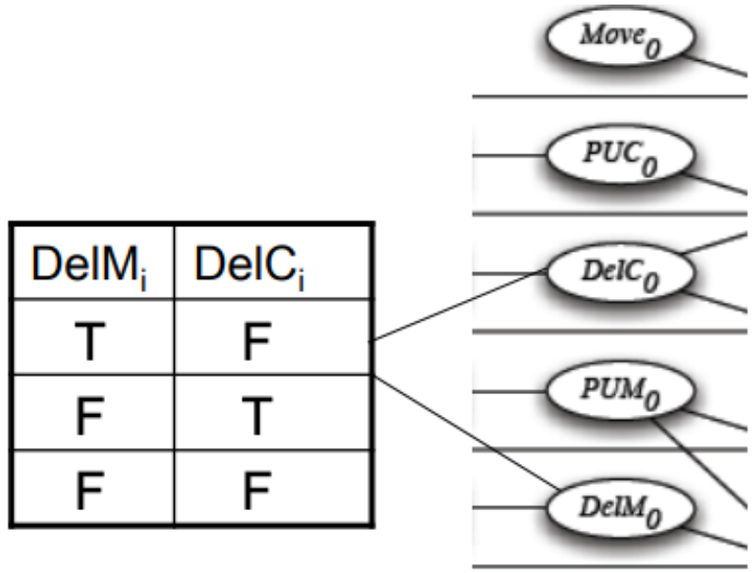
- explain how a state variable at time $t+1$ is affected by the action(s) taken at time t and by its own value at time t

RHC_t	$DelC_i$	PUC_i	RHC_{t+1}
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

CSP planning: additional constraints

Other constraints we may want are **action constraints**:

- specify which actions **cannot occur simultaneously**
- these are sometimes called **mutual exclusion** (mutex) constraints

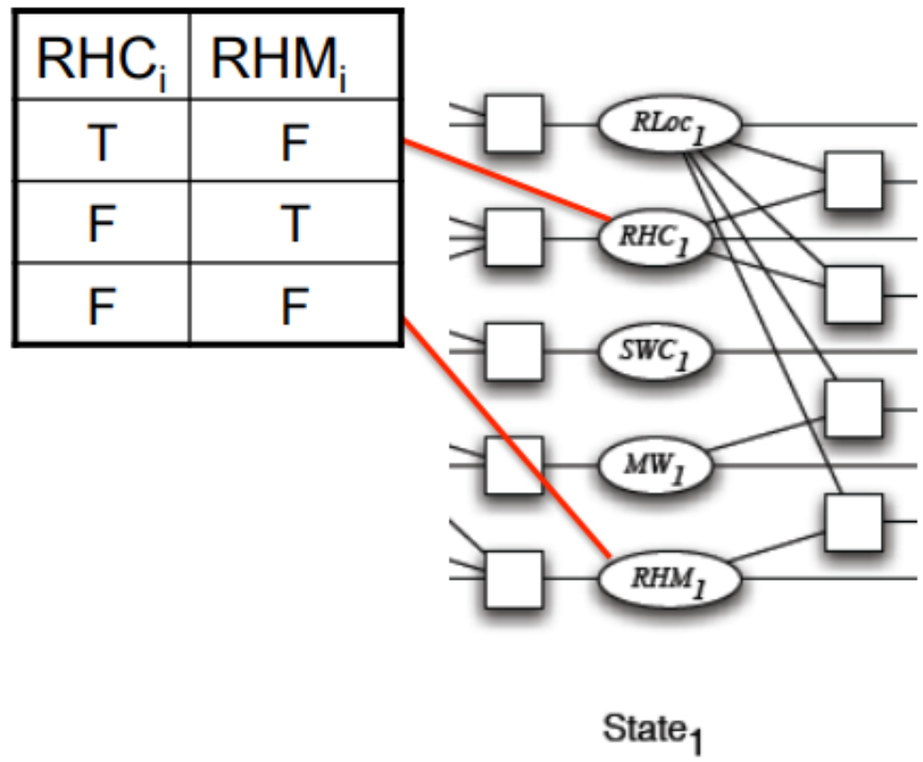


Example: in the Robot domain DelM and DelC can occur in any sequence (or simultaneously). But we can enforce that they do not happen simultaneously

CSP planning: additional constraints

Other constraints we may want are **state constraints**

- hold between variables at the same time step
- they can capture physical constraints of the system



Example: robot cannot hold coffee and mail

CSP planning: Solving the problem

We need to "unroll the plan" for a fixed number of steps: this is called the **horizon**.

Map STRIPS Representation into CSP for horizon $k = 1,2,3,\ldots$ Solve CSP for horizon $k = 1,2,3,\ldots$ until solution found at the lowest possible horizon.

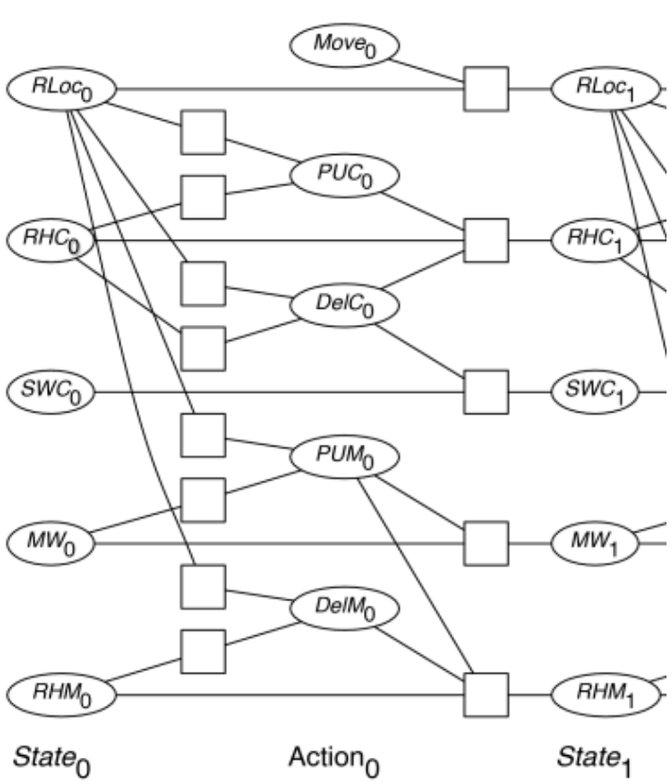
Example: $k = 0$
Is $State_0$ a goal?

If yes, DONE! If no,

CSP planning: Solving the problem

Map STRIPS Representation into CSP for horizon $k = 1$

Solve CSP for horizon $k = 1$



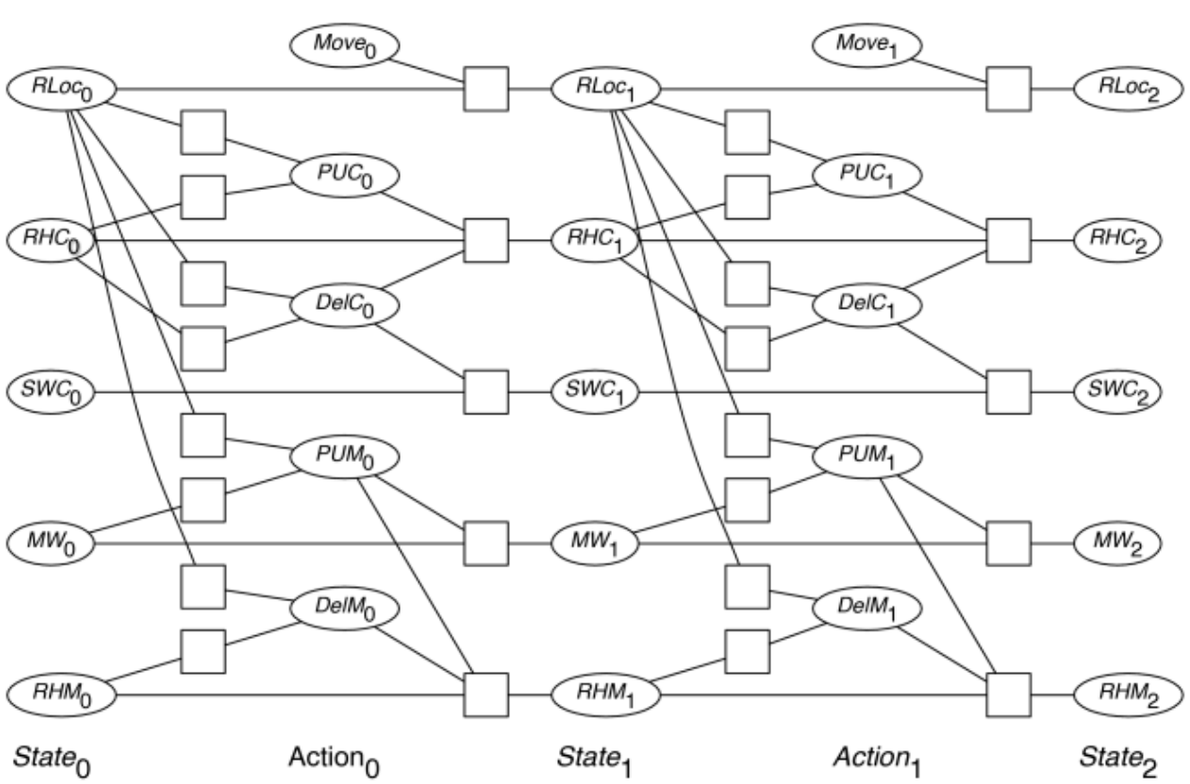
Example: $k = 1$
Is $State_1$ a goal?

If yes, DONE! If no, continue

CSP planning: Solving the problem

Map STRIPS Representation into CSP for horizon $k = 2$

Solve CSP for horizon $k = 2$



Example: $k = 2$
Is $State_2$ a goal?

If yes, DONE! If no, continue

Solve planning as CSP: Pseudo code

```
solved = false
horizon = 0
while solved = false
      map STRIPS into CSP with horizon
      solve CSP → solution
      if solution then
         solved = T
      else
         horizon = horizon + 1
Return solution
```

Planning as CSP

If the algorithm for planning as CSP stops and returns a solution plan of length k, does it mean that there are no shorter solutions?

A. Yes

B. No

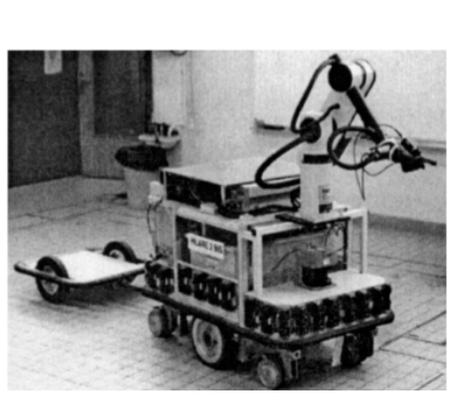
C. It depends

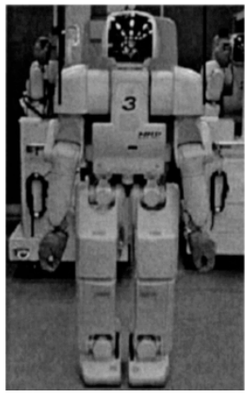
STRIPS to CSP applet

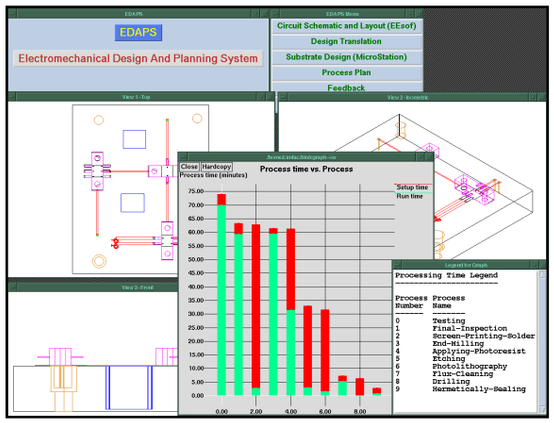
Allows you:

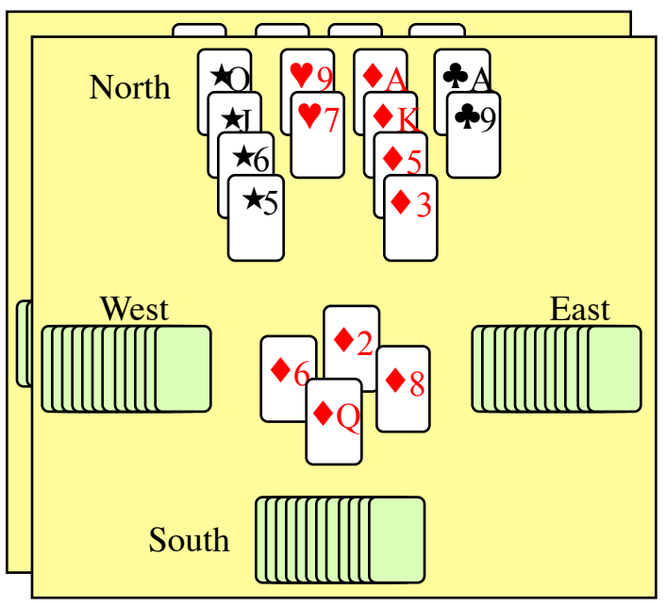
- to specify a planning problem in STRIPS
- to map it into a CSP for a given horizon
- the CSP translation is automatically loaded into the CSP applet where it can be solved
- Under "Main Tools" in the AIspace Home Page

Some applications of planning









Emergency Evacuation?

Robotics?

Space Exploration?

Manufacturing Analysis?

Games (e.g., Bridge)?

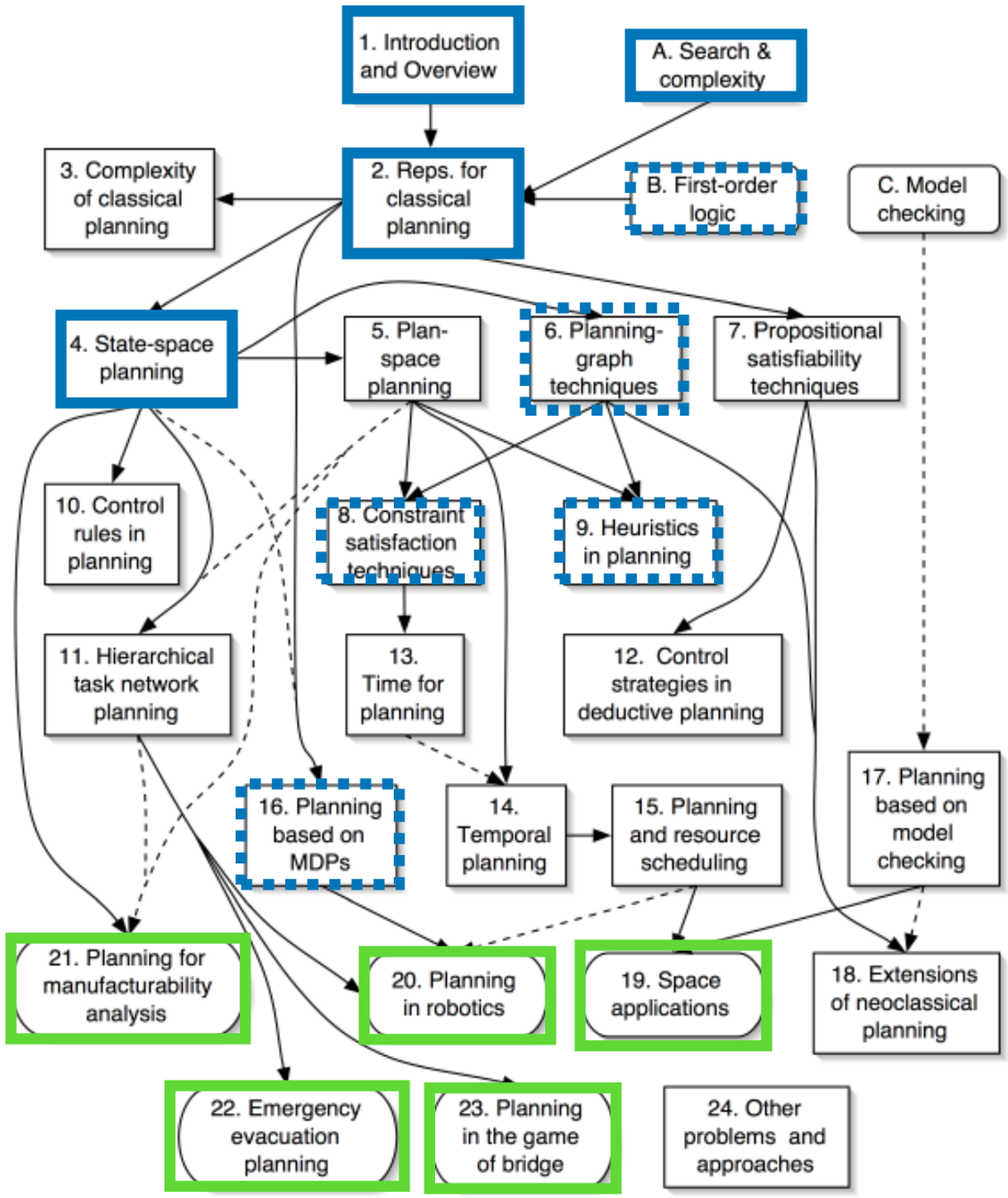
Generating Natural language

Product Recommendations ….

You (will) know the key ideas 🙂

You know

Know little



Ghallab, Nau, and Traverso

Automated Planning: Theory and Practice, Morgan Kaufmann, May 2004

Web site:
http://www.laas.fr/planning

http://projects.laas.fr/planning/

Applications

Practice exercises

http://www.aispace.org/exercises/exercise8-a-1.shtml
http://www.aispace.org/exercises/exercise8-b-1.shtml

A rough CPSC 322 overview

Representation and reasoning

Environment

Deterministic | Stochastic

Problem

Static | Sequential

Constraint satisfaction | Query

Planning

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

Coming up: Logics

Mostly only propositional….This is the starting point for more complex ones ….

Natural to express knowledge about the world

What is true (boolean variables)

How it works (logical formulas)

Well understood formal properties

Boolean nature can be exploited for efficiency