

Planning: Heuristics and CSP Planning

Computer Science cpssc322, Lecture 18
(Textbook Chpt 8)

Oct, 18, 2013



Lecture Overview

- **Recap: Planning Representation and Forward algorithm**
- Heuristics
- CSP Planning



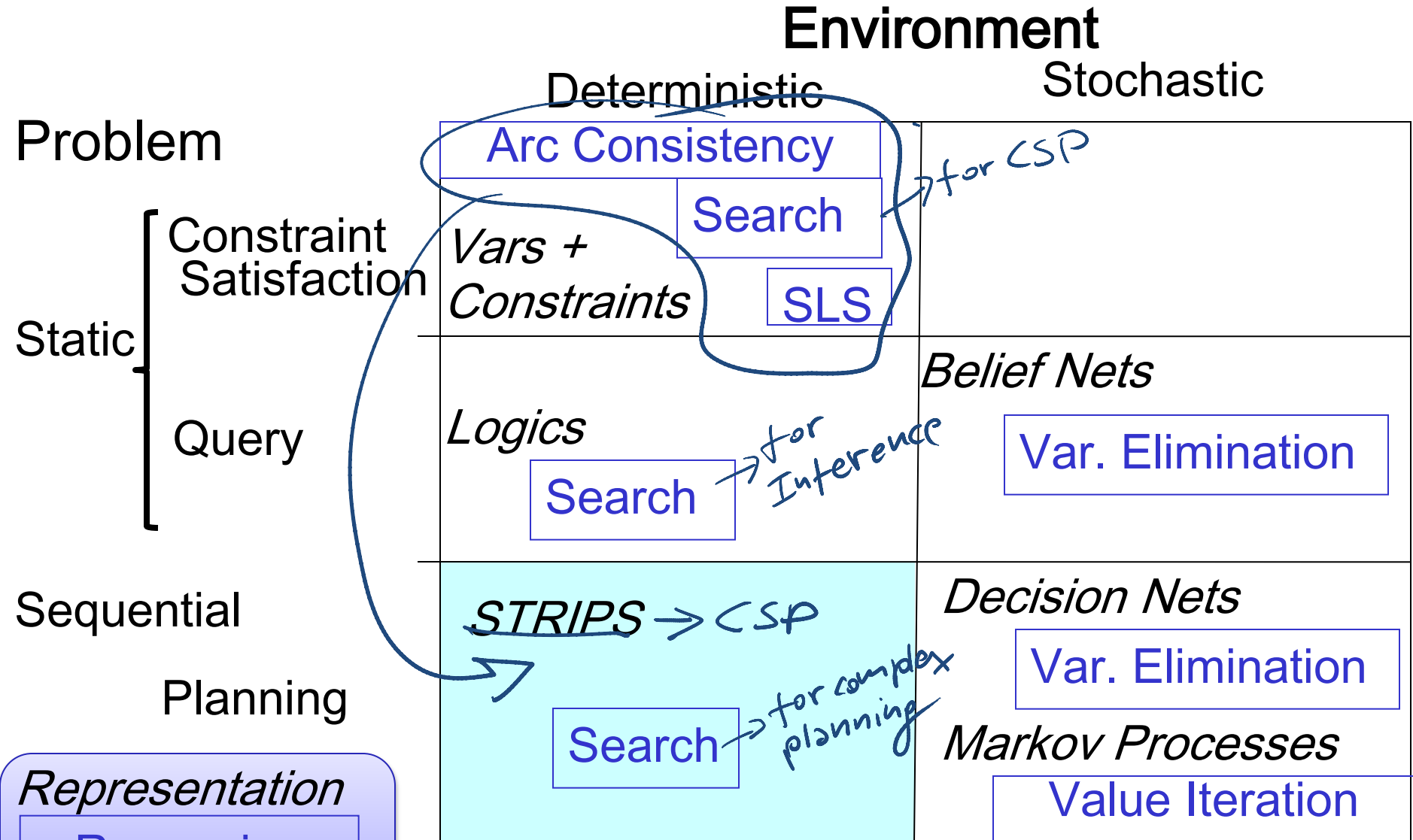
A.

B.

C.

D.

Modules we'll cover in this course: R&Rsys



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?** A. Full assignment B. Partial assignment
- **Successor function?**
- **Goal test?** A. Full assignment B. Partial assignment
- **Solution?**
- **Heuristic function....**

Inference

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

Lecture Overview

- Recap: Planning Representation and Forward algorithm
- Heuristics for forward planning ←
- CSP Planning

Heuristics for Forward Planning

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

In planning this is the...# actions

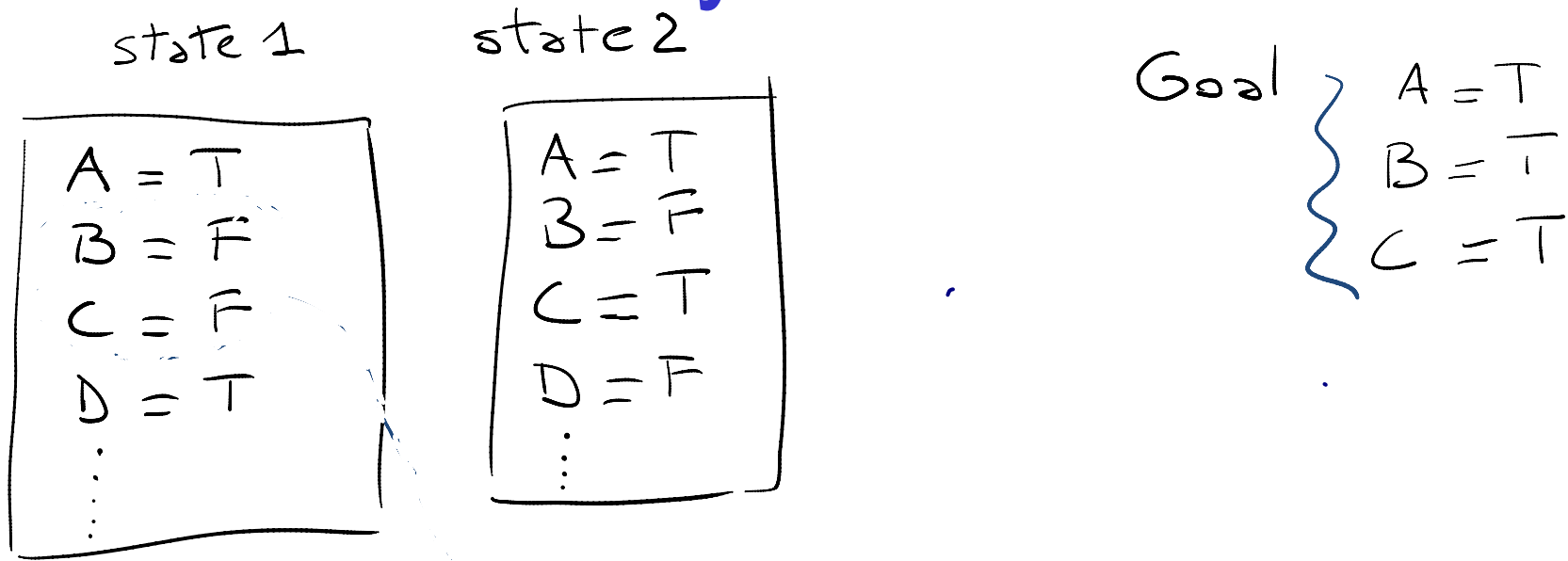
Two simplifications in the representation:

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

And a Def. a **subgoal** is a particular assignment in the goal e.g., if the goal is <A=T, B=T, C=T> then....

subgoal A=T subgoal B=T subgoal C=T

Heuristics for Forward Planning: Any ideas?



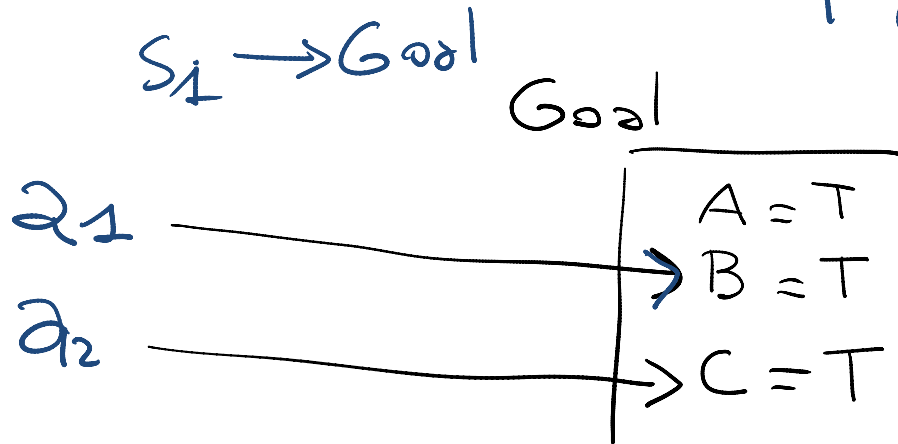
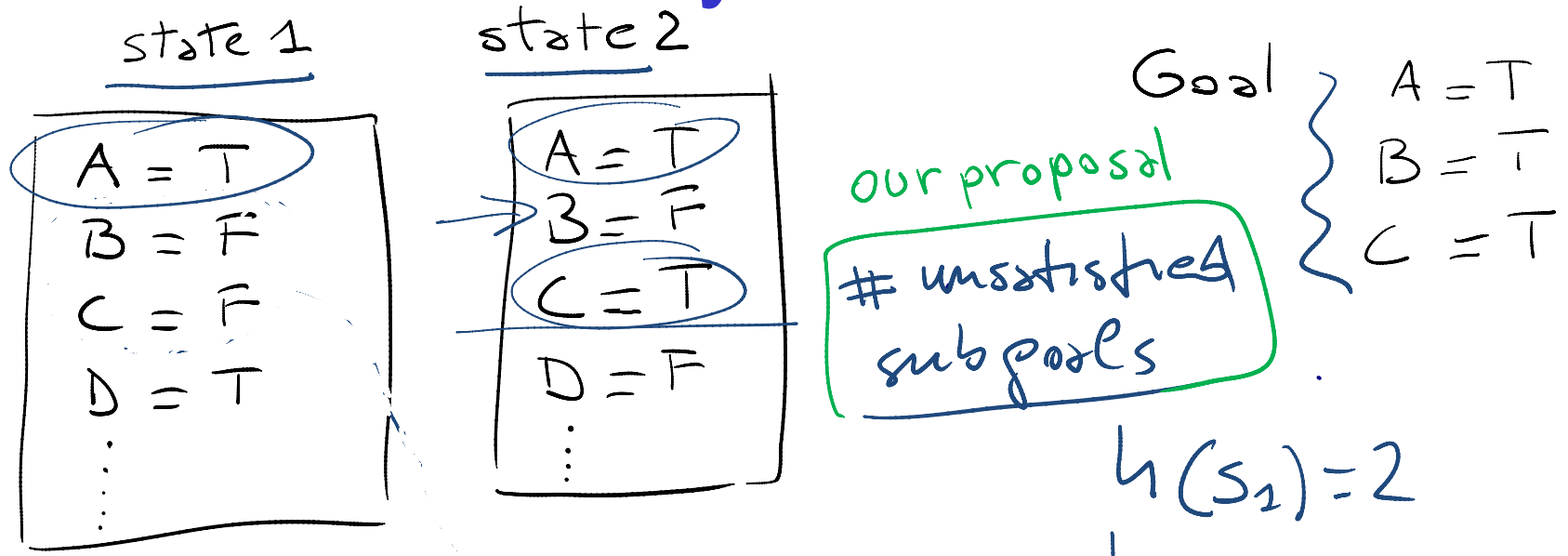
A. Number of satisfied sub-goals



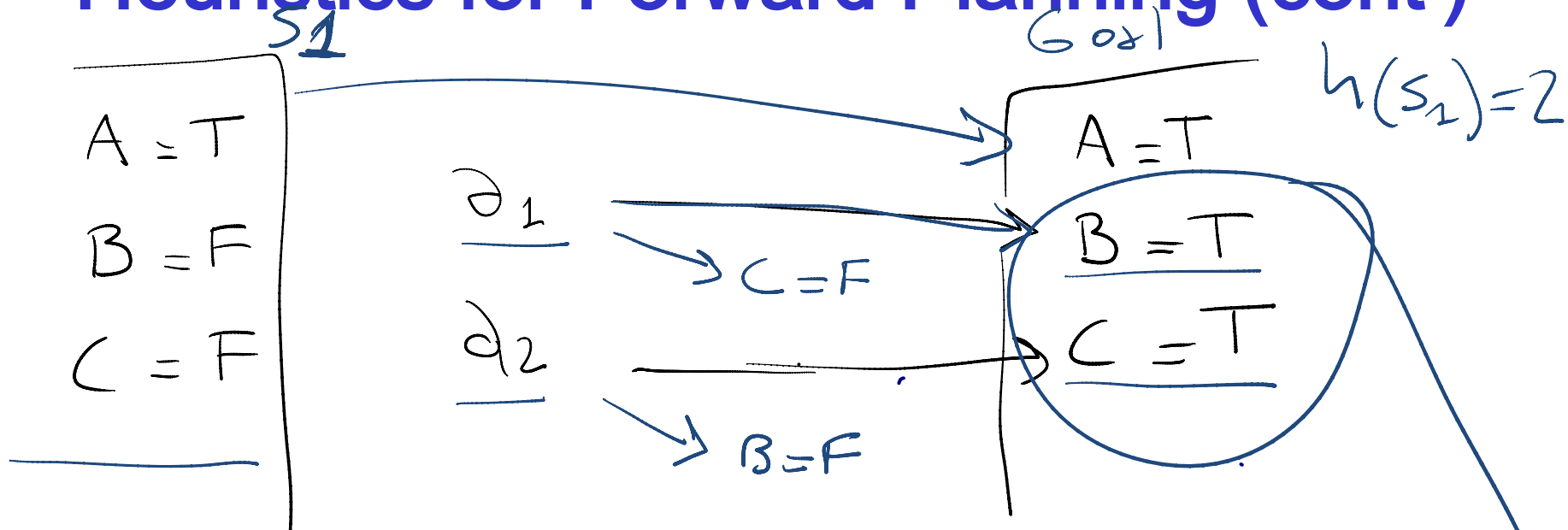
B. Number of unsatisfied sub-goals

C. None of the above

Heuristics for Forward Planning: Any ideas?



Heuristics for Forward Planning (cont')



What kind of simplifications of the actions would justify our proposal for h ?

- ~~→ a) We have removed all preconditions *too strong!* VERY STRONG~~
- b) We have removed all "negative" effects**
- ~~→ c) We assume no action can achieve both INADMISSIBLE~~

Heuristics for Forward Planning: empty-delete-list

So

- We only relax the problem according to (...*b*...)

i.e., we remove all the effects that make a variable F

Action *a* effects (~~B=F~~, C=T)

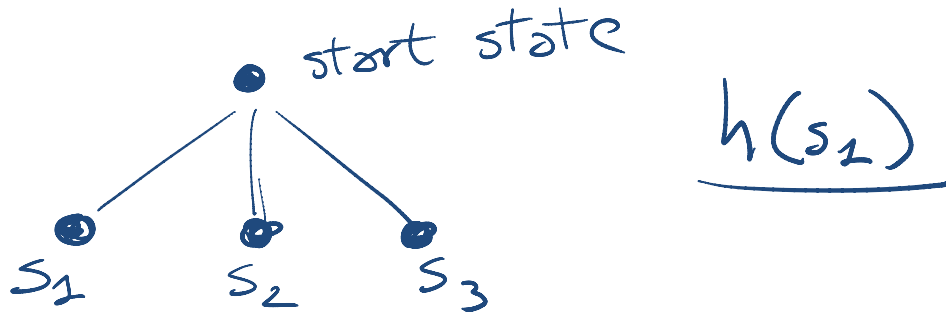
- But then how do we compute the heuristic? ↙

..... solve a simplified planning prob.

This is often fast enough to be worthwhile

- empty-delete-list heuristics with forward planning is currently considered a very successful strategy

Empty-delete in practice



to compute $h(s_i)$, run forward planner with s_i as start state, with the same goal as the original problem but with all the actions with the negative effects removed.

So to compute h \rightarrow for a given state we need to solve a planning problem (but a simpler one!)
You may need to do this MANY times \leftarrow


Final Comment

- 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
 - See, for instance, descriptions of competing planners in the presentation of results for the **2008 planning competition** (posted in the class schedule)

Lecture Overview

- Recap: Planning Representation and Forward algorithm
- Heuristics for forward planning
- **CSP Planning**

Planning as a CSP

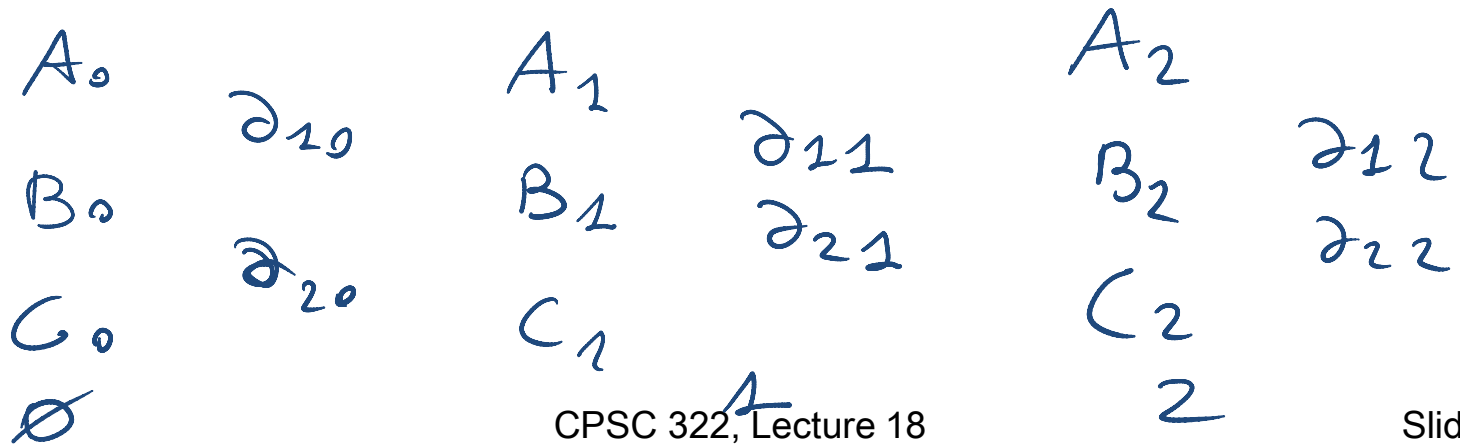
- An alternative approach to planning is to set up a planning problem as a CSP!
 - We simply reformulate a STRIPS model as a set of variables and constraints
 - Once this is done we can even express additional aspects of our problem (as additional constraints)
- e.g., see [Practice Exercise](#) UBC commuting
“*careAboutEnvironment*” constraint 

Planning as a CSP: Variables

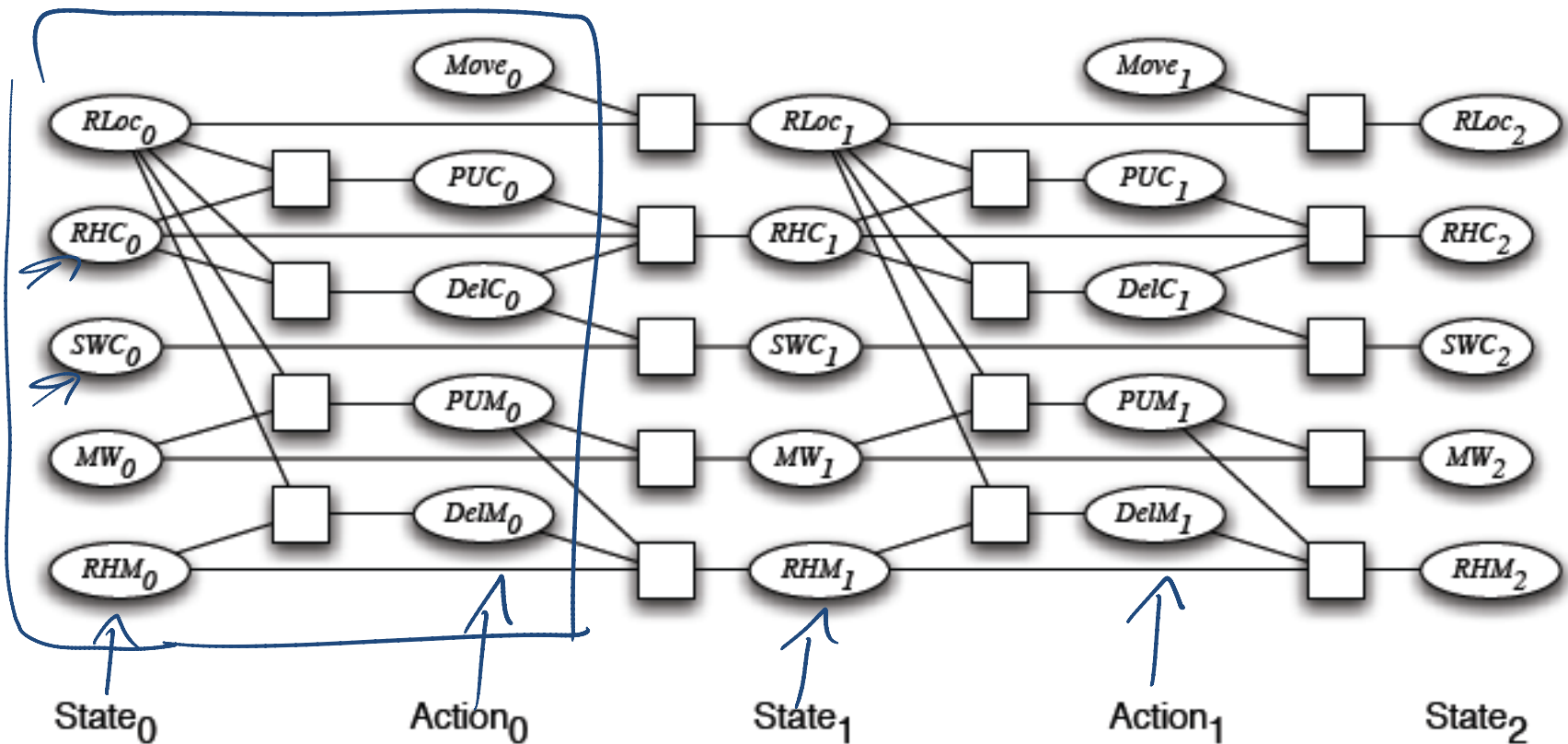
- We need to “unroll the plan” for a fixed number of steps: this is called the **horizon**
- To do this with a horizon of k :
 - construct a **CSP variable** for each **STRIPS variable** at each time step from 0 to k
 - construct a **boolean CSP variable** for each **STRIPS action** at each time step from 0 to $k - 1$.

A B C

∂_1
 ∂_2



CSP Planning: Robot Example

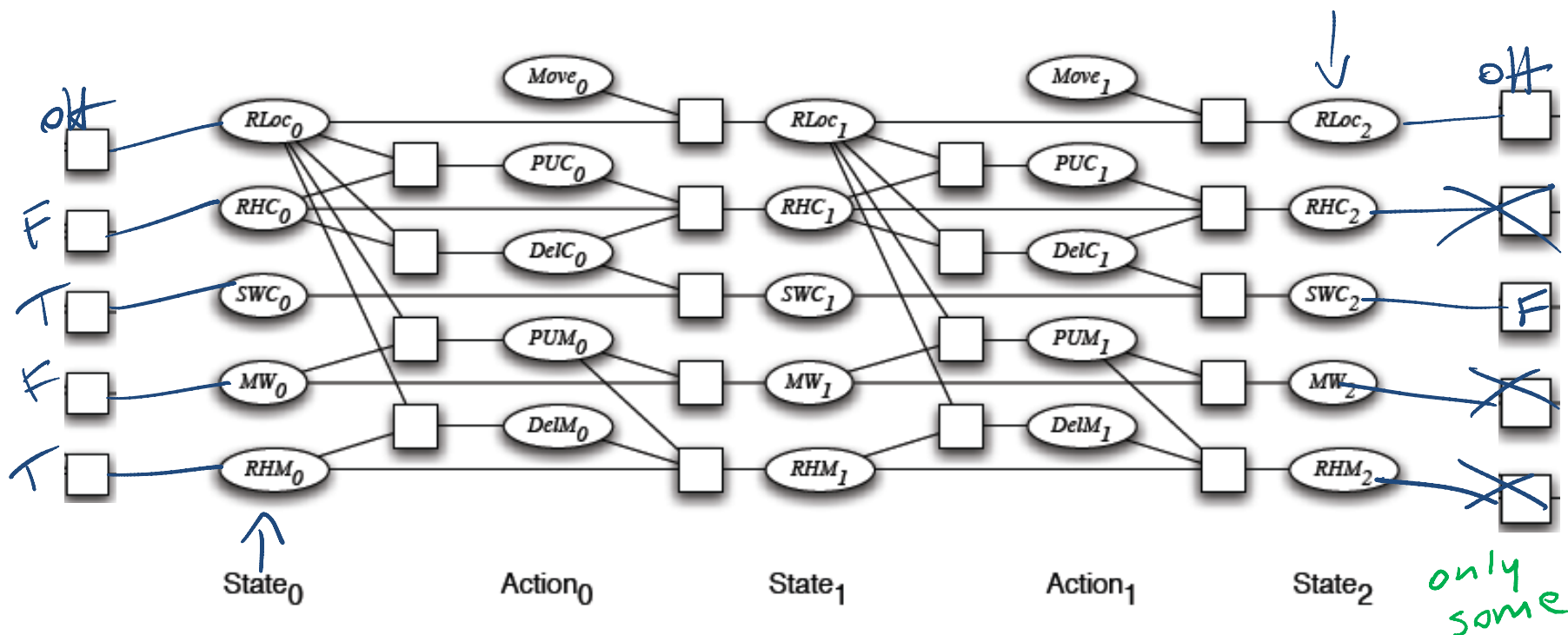


Variables for actions binary

action (non) occurring at that step

CSP Planning: Initial and Goal Constraints

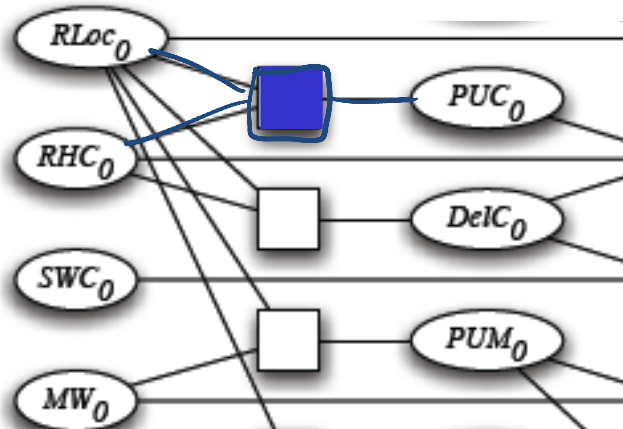
- 05255
- initial state constraints constrain the state variables at time 0
 - goal constraints constrain the state variables at time k



CSP Planning: Prec. 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



Rob Location →

Rob has coffee →

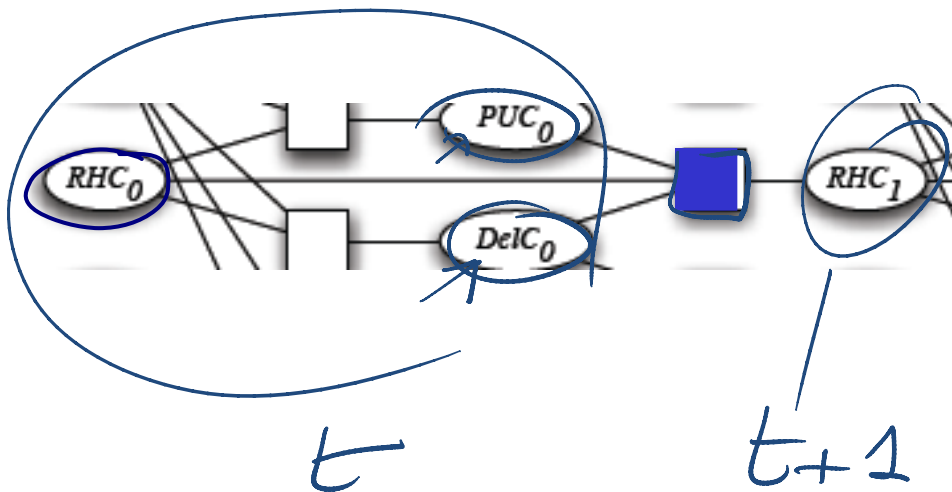
	RLoc ₀	RHC ₀	PUC ₀
CS		T	<u>F</u>
CS		F	T
CS		F	<u>F</u>
mr		*	<u>F</u>
lab		*	<u>F</u>
off		*	<u>F</u>

PUC₀ (pick up coffee) →

CSP Planning: Effect 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_i	$DelC_i$	PUC_i	RHC_{i+1}
T	T	T	<u>T</u>
T	T	F	F
T	F	<u>T</u>	T
...
...

CSP Planning: Constraints Contd.

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

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



How can we specify that $DelM$ and $DelC$ cannot occur simultaneously ?

A.

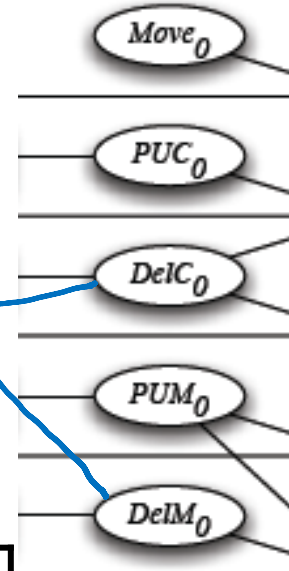
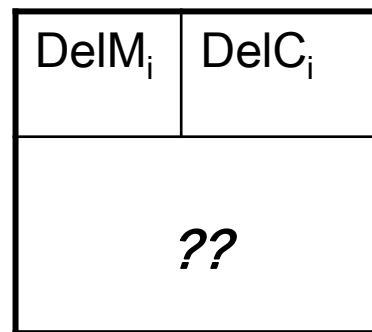
$DelM_i$	$DelC_i$
T	T
T	F
F	T

B.

$DelM_i$	$DelC_i$
T	T
F	F

C.

$DelM_i$	$DelC_i$
T	F
F	T
F	F



Action₀

CSP Planning: Constraints Contd.

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

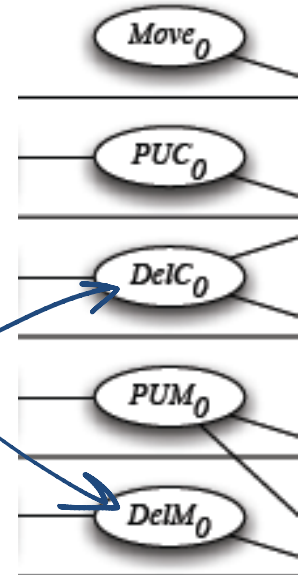
- specify which actions cannot occur simultaneously
- these are sometimes called mutual exclusion (mutex) constraints

E.g., in the Robot domain

DelM and *DelC* can occur in any sequence (or simultaneously)

But we could change that...

$DelM_i$	$DelC_i$
T	F
F	T
F	F

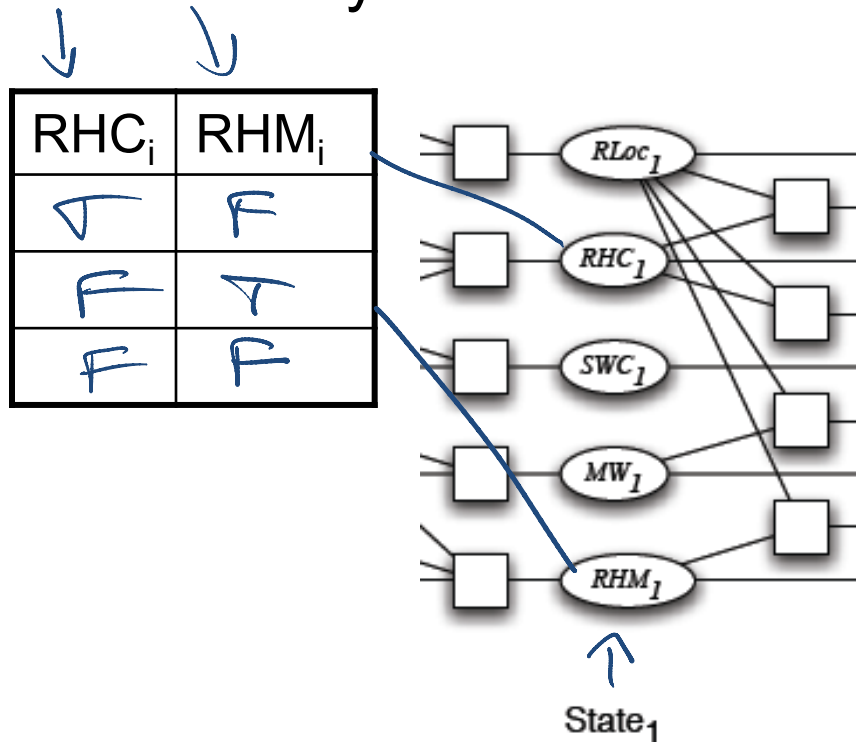


$Action_0$

CSP Planning: Constraints Contd.

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

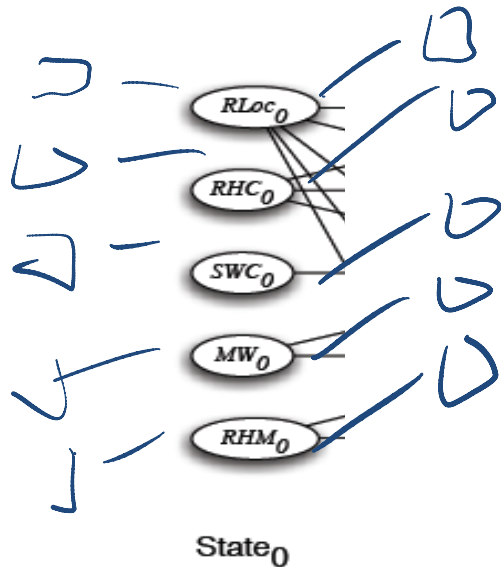
- hold between variables at the same time step
- they can capture physical constraints of the system (robot cannot hold coffee and mail)
- they can encode maintenance goals



CSP Planning: Solving the problem

Map STRIPS Representation for horizon 1, 2, 3, ..., until solution found

Run arc consistency and search!



$k = 0$

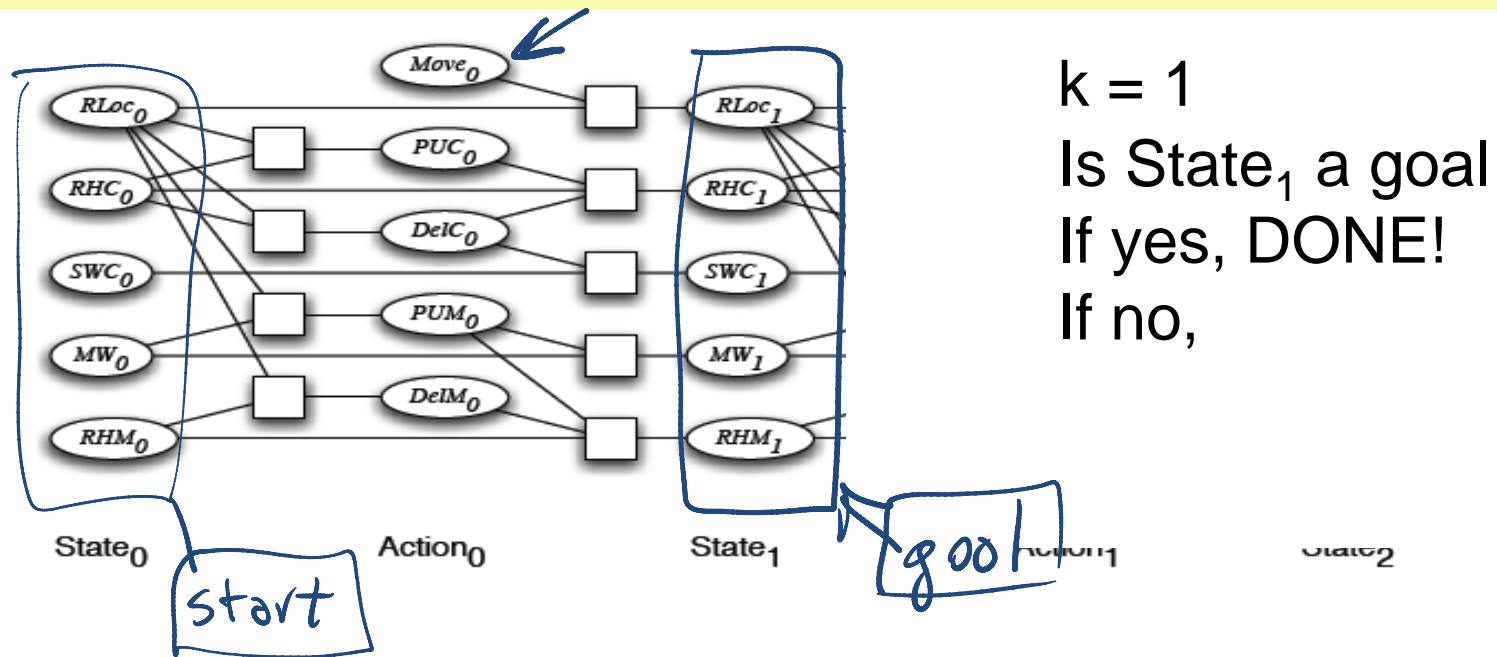
Is $State_0$ a goal?

If yes, DONE!

If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon $k = 1$
Run arc consistency and search!

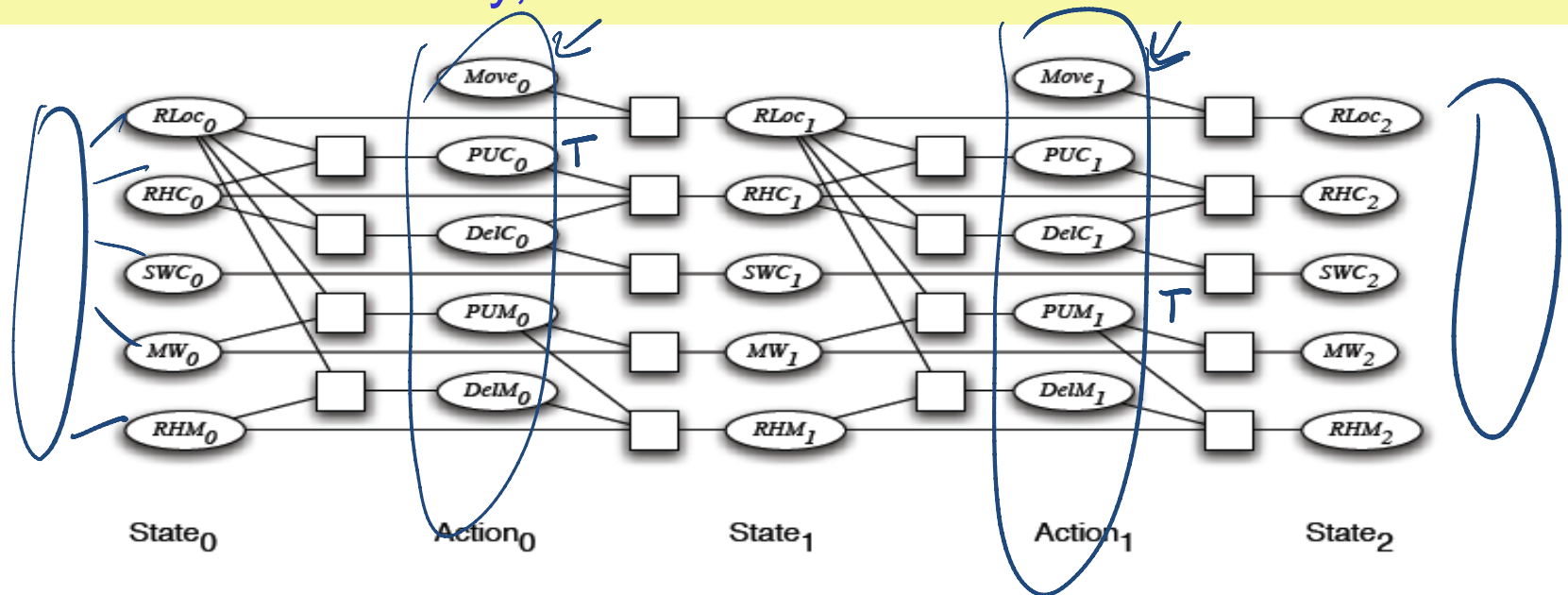


$k = 1$
Is State₁ a goal
If yes, DONE!
If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon $k = 2$

Run arc consistency, search!



$k = 2$: Is State₂ a goal
If yes, DONE!
If no....continue

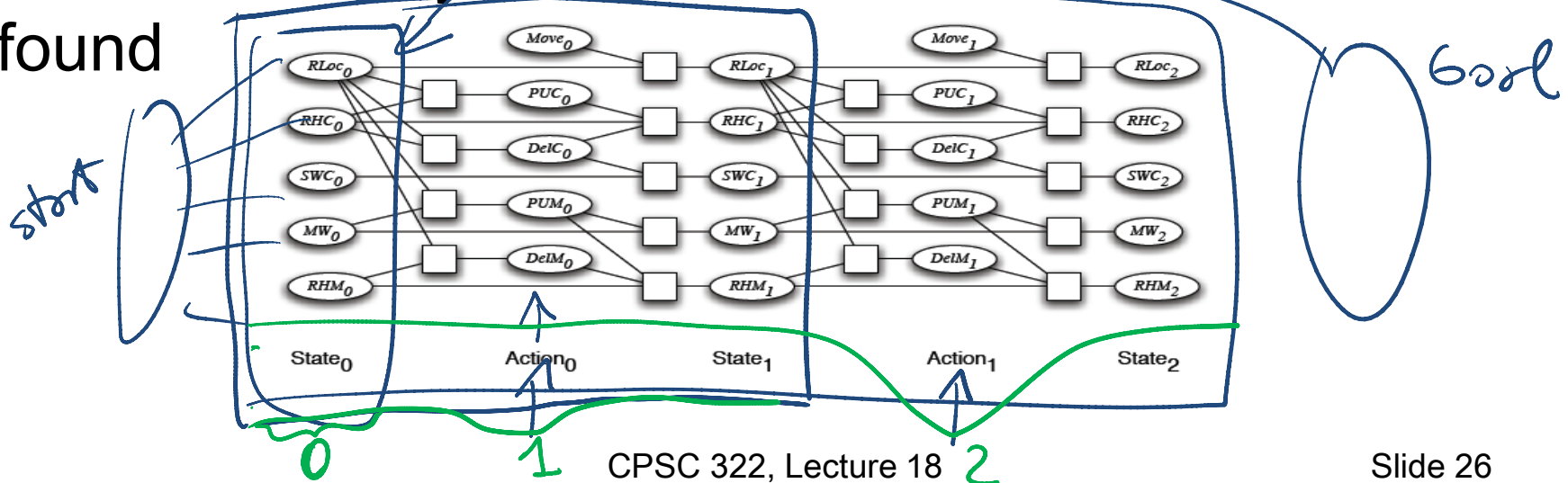
CSP Planning: Solving the problem

Map STRIPS Representation for horizon: 0 1 2 ...

Run arc consistency and search,

Plan: all actions with assignment T

In order to find a plan, we expand our constraint network one layer at the time, until a solution is found



Solve planning as CSP: pseudo code

solved = false
horizon = 0

while not solved

→ map STRIPS to CSP with horizon

→ solve CSP → solution

if solution found then

solved = true

else

horizon = horizon + 1

return solution

State of the art planner

A similar process is implemented (more efficiently) in the **Graphplan** planner



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

Practice exercise using STRIPS to CSP is available on AIspace

Learning Goals for today's class

You can:

- Construct and justify a **heuristic function** for forward planning.
- Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
- Solve a planning problem with CPS by expanding the horizon (new one)

322 Feedback 😊 or 😞

- Lectures
- Slides
- Practice Exercises
- Assignments
- Alspace
- Clickers
- Textbook
- Course Topics / Objectives
- TAs
- Learning Goals
-

What is coming next ?

Textbook Chpt 5.1-5.1.1 – 5.2

Environment

Deterministic

Stochastic

Problem

Constraint Satisfaction

Vars + Constraints

Search

SLS

for CSP

Static

Inference

Logics

Search

CSP for Inference

Belief Nets

Var. Elimination

Sequential

Planning

STRIPS

CSP

Search

for complex planning

Decision Nets

Var. Elimination

Markov Processes

Value Iteration

Representation

Reasoning
Technique

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
-

Thxs for the honest Feedback: Most mentioned issues

- Confusion about what is the right answer to questions (including card ones) – see inked slides for unambiguous answer. Typically one slides has the question the next one has the answer. Please let me know if any is missing
- Flash cards vs. iClickers (private)
- Samples for midterm