Introduction to
Artificial Intelligence (AI)

Computer Science cpsc502, Lecture 5

Sep, 22, 2011

Today Sept 22

- **Finish Stochastic Local Search (SLS)**
- Planning
  - STRIPS
  - Heuristics
  - STRIPS -> CSP

Population Based SLS

Often we have more memory than the one required for current node (+ best so far + tabu list)

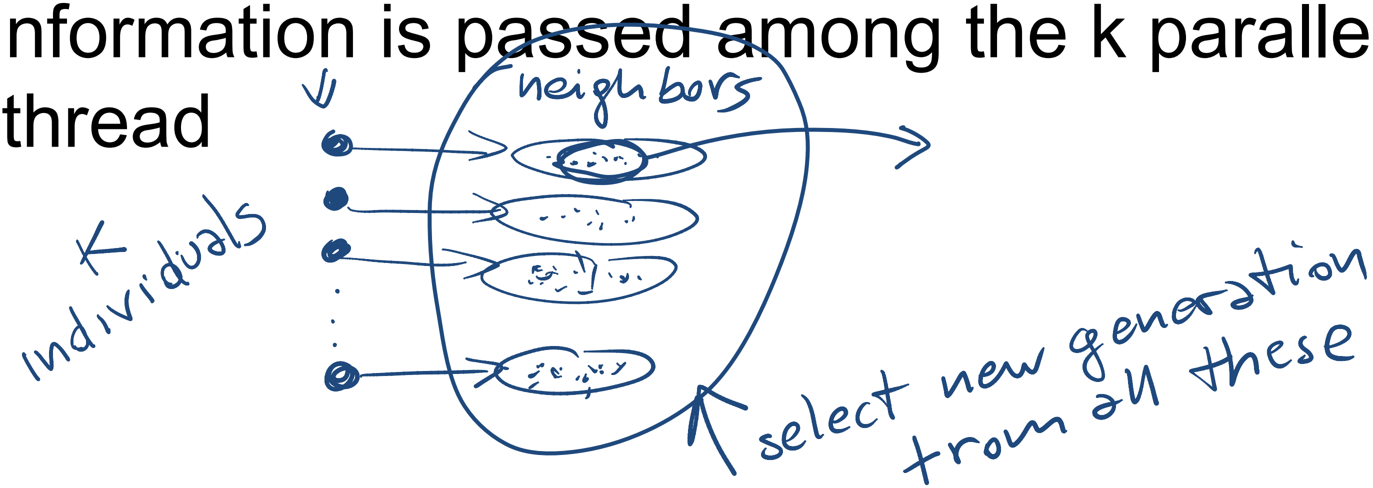
Key Idea: maintain a population of *k* individuals

- At every stage, update your population.
- Whenever one individual is a solution, report it.

Population Based SLS: Beam Search

Non Stochastic

- Start with *k* individuals, and choose the *k* best out of all of the neighbors.
- Useful information is passed among the k parallel search thread



- **Troublesome case:** If one individual generates several good neighbors and the other k-1 all generate bad successors…. the next generation will comprise very similar individuals :(

Population Based SLS: Stochastic Beam Search

- **Non Stochastic** Beam Search may suffer from lack of diversity among the k individual (just a more expensive hill climbing)
- **Stochastic** version alleviates this problem:
  - Selects the k individuals at random
  - But probability of selection proportional to their value (according to scoring function)

$\bigcirc\!\!\!\!\!m$ neighbors $\underbrace{\{n_1 \dots n_m\}}$

$\underline{h}$: scoring function

$$\underline{\text{Probability of selecting}}\,(\underset{\uparrow}{n_J}) = \frac{h(n_J)}{\sum_{i=1}^{m} h(n_i)}$$

Stochastic Beam Search: Advantages

- It **maintains diversity** in the population.
- **Biological metaphor** (asexual reproduction):
  - ✓ each individual generates "mutated" copies of itself (its neighbors)
  - ✓ The scoring function value reflects the fitness of the individual
  - ✓ the higher the fitness the more likely the individual will survive (i.e., the neighbor will be in the next generation)

Population Based SLS: Genetic Algorithms

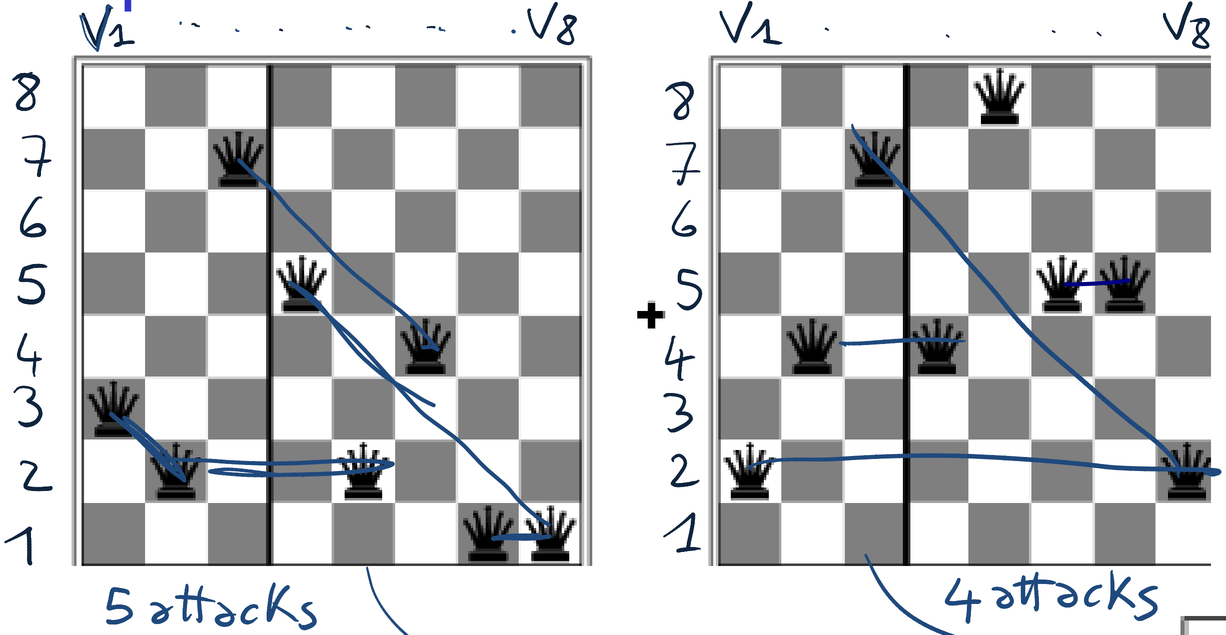
- Start with *k* randomly generated individuals (population)
- An individual is represented as a string over a finite alphabet (often a string of 0s and 1s)
- A successor is generated by combining two parent individuals (loosely analogous to how DNA is spliced in sexual reproduction)
- Evaluation/Scoring function (fitness function). Higher values for better individuals.
- Produce the next generation of individuals by selection, crossover, and mutation

Genetic algorithms: Example 8-queen

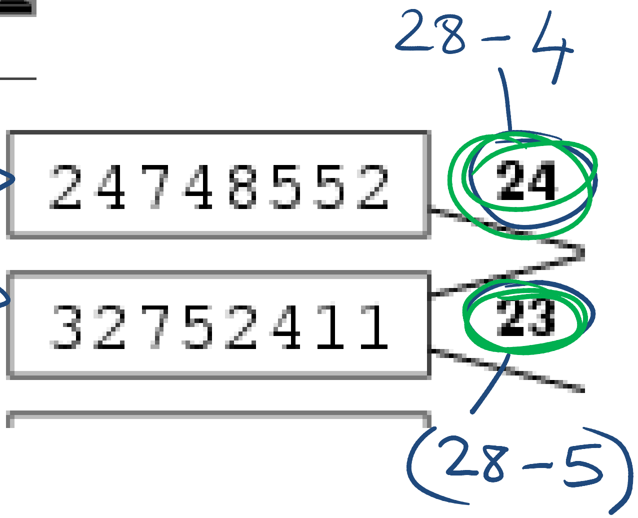
Representation and fitness function

of queen pairs possibly attacking each other

$\frac{8 \cdot 7}{2} = 28$



State: string over finite alphabet

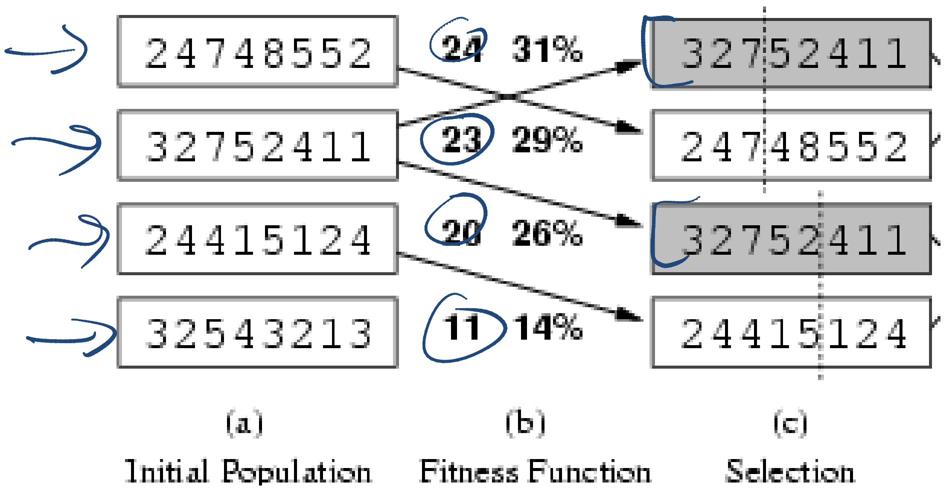


Fitness function: higher value better states.

queen pairs not attacking each other

Genetic algorithms: Example

Selection: common strategy, probability of being chosen for reproduction is directly proportional to fitness score



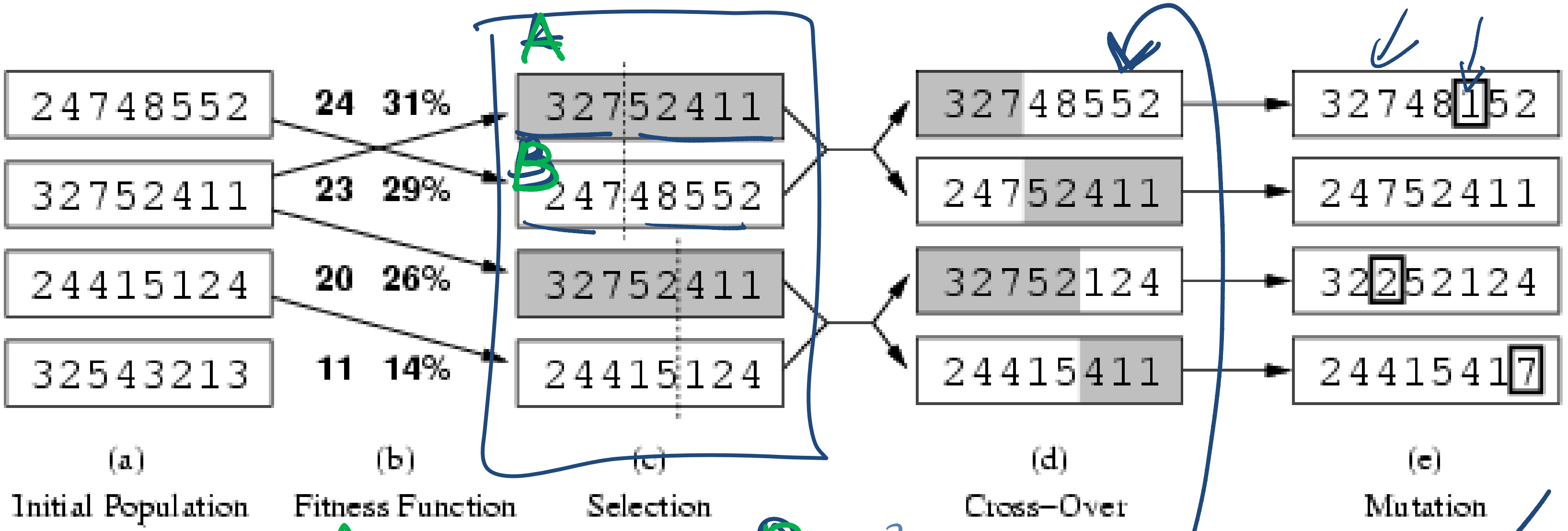
24/(24+23+20+11) = 31%

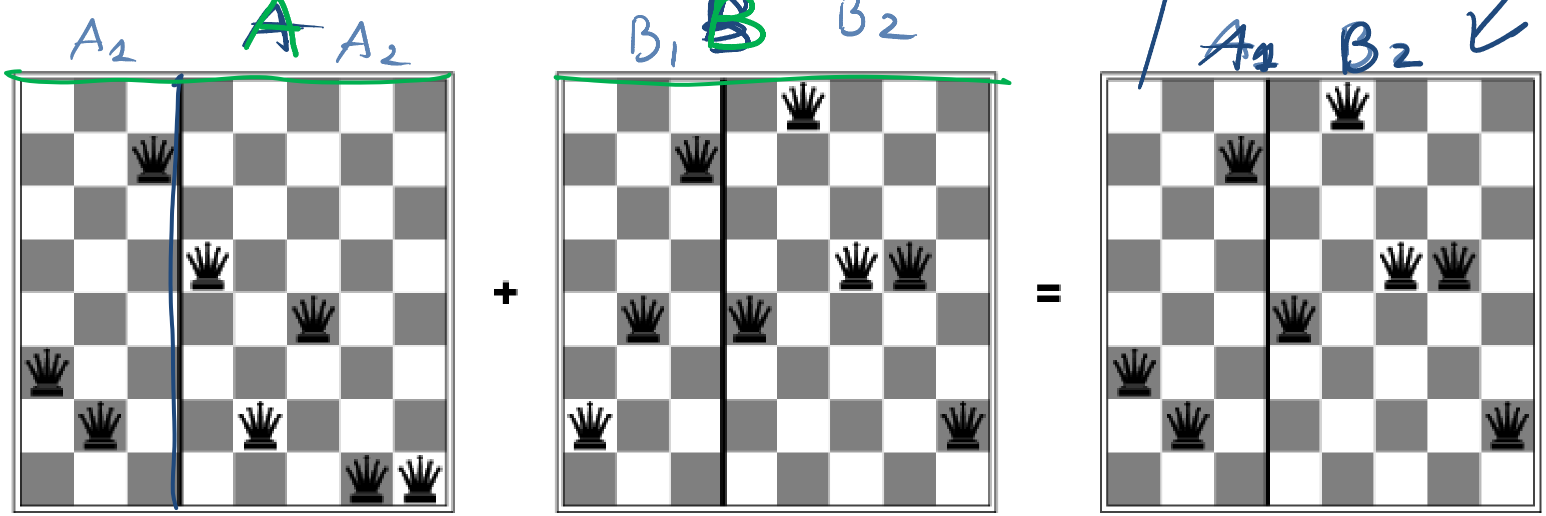
23/(24+23+20+11) = 29% etc

same as Beam Search

Genetic algorithms: Example

Reproduction: cross-over and mutation





Genetic Algorithms: Conclusions

- Their performance is very sensitive to the choice of state representation and fitness function
- **Extremely slow** (not surprising as they are inspired by evolution!)

- But relatively simple example of biologically inspired AI approaches

Today Sept 22

- Finish Stochastic Local Search (SLS)
- **Planning**
  - STRIPS – Forward Planning
  - Heuristics
  - STRIPS -> CSP

R&Rsys we'll cover in this course

Environment

Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency SLS Search	
Static	Query	*Logics* → Propositional → First Order → . . . . Search	*Belief Nets* Var. Elimination Approx. Inference Temporal. Inference
Sequential	Planning	*STRIPS* actions precs effects Search	*Decision Nets* aka influence diagrams Var. Elimination *Markov Processes* Value Iteration

Representation

Reasoning Technique

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
- Successor function
- Goal test
- Solution
- Heuristic function

Inference

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

Planning as Search: State and Goal

How to **select and organize a sequence of acti**ons to **achieve a given goal**…

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

A B C domain(true, false) (T, F)

(1 0)

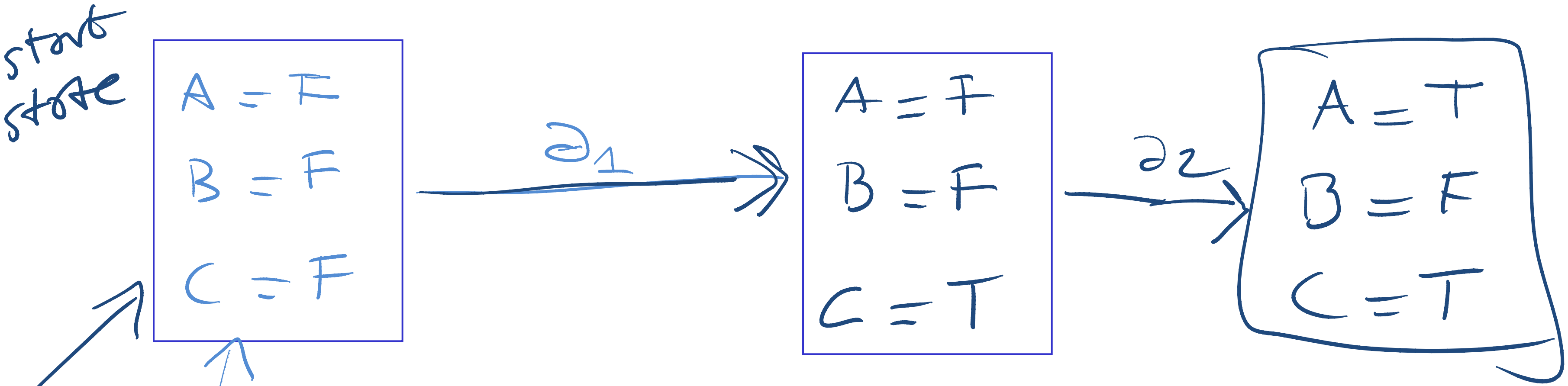
$$\begin{bmatrix} A = T \\ B = F \\ C = T \end{bmatrix}$$

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

A = T C = F

Planning as Search: Successor function and Solution

Actions : take the agent from one state to another



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

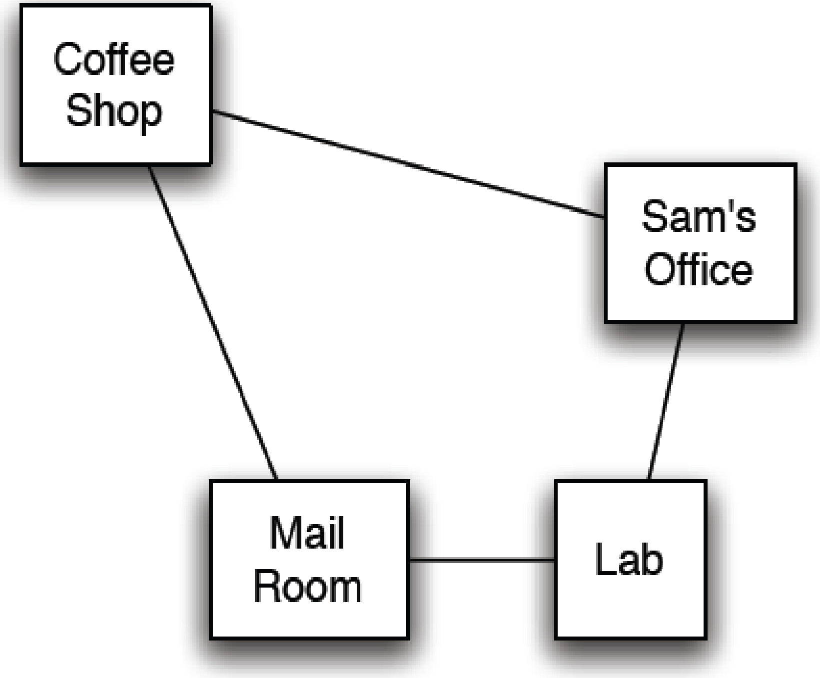
sol a_1 a_2

A = T

Goal

Delivery Robot Example (textbook)

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



For another example see Practice Exercise 8.C: "Commuting to UBC"

Delivery Robot Example: States

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=T RHC=F

RHC - Rob has coffee True/False. → rhc $\overline{rhc}$

SWC - Sam wants coffee T/F

MW - Mail is waiting T/F

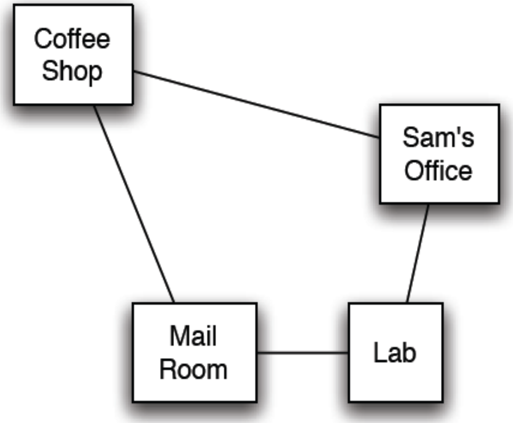
RHM - Rob has mail T/F

4 2 2 2 2

Example state: $\{cs, rhc, \overline{swc}, \overline{mw}, rhm\}$

Number of states: 64

Delivery Robot Example: Actions



The robot's actions are:

Move - Rob's move action

- move clockwise (*mc*), move anti-clockwise (*mac*) ~~not move (nm)~~

PUC - Rob picks up coffee

- must be at the coffee shop

preconditions

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

STanford Research Institute Problem Solver (STRIPS) action representation

The key to sophisticated planning is modeling actions

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: Examples

STRIPS representation of the action pick up coffee, *PUC* :

- **preconditions** *Loc* = *cs* and *RHC* = F
- **effects** *RHC* = T

STRIPS representation of the action deliver coffee, *DelC* :

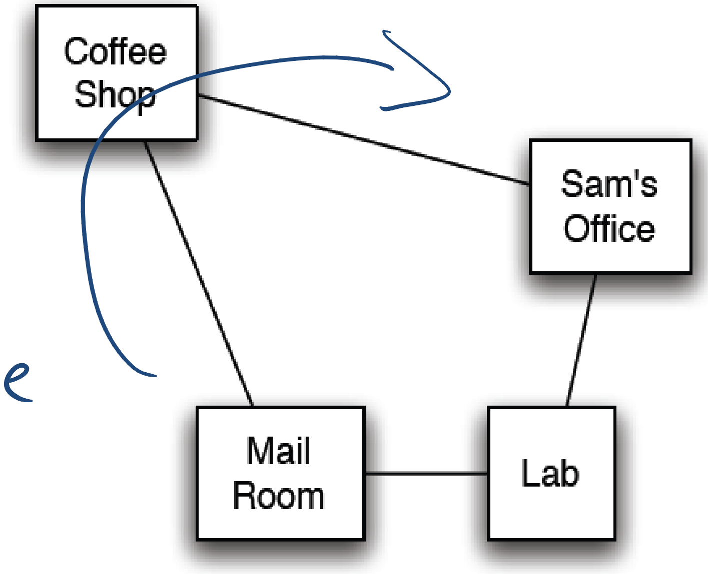
- **preconditions** *Loc* = off and *RHC* = T (SWC = T)
- **effects** RHC = F and SWC = F

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: MC and MAC

STRIPS representation of the action

MoveClockwise ?



Prec loc=CS

mc-cs

Eff loc=office

mc-off

⋮

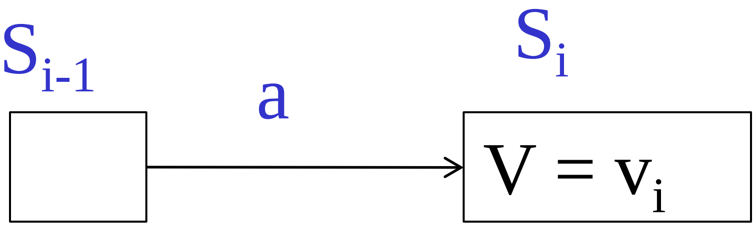
mac-cs

STRIPS Actions (cont')

The STRIPS assumption:

all features/variables not explicitly changed by an action stay unchanged

- So if the feature V has value v_i in state 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?



Forward Planning

To find a plan, a solution: search in the state-space graph.

- The **states** are the **possible worlds**
- The **arcs** correspond to the **actions**: The arcs from a state ***s*** represent all of the actions that are legal in state ***s***. *(What actions are legal?)*

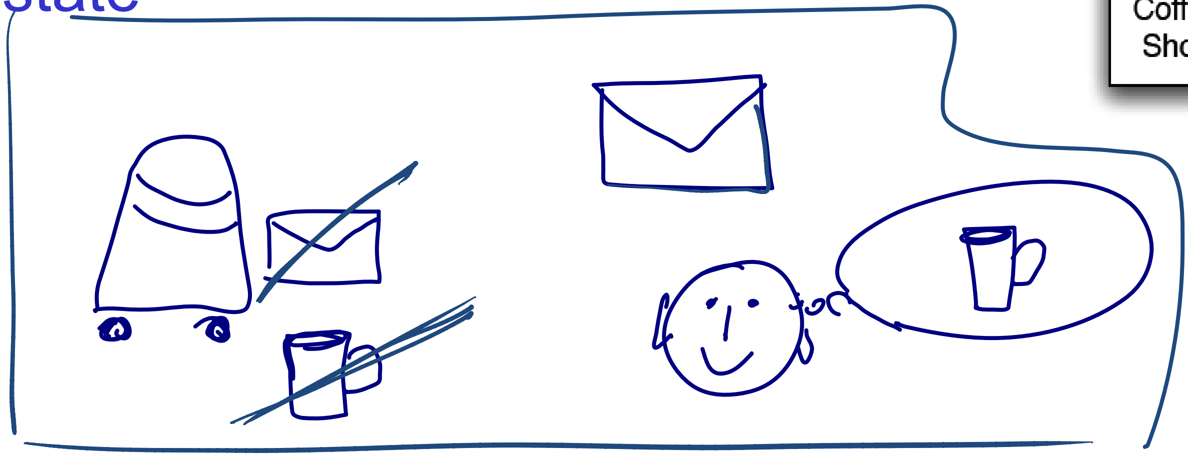
whose preconds are satisfied

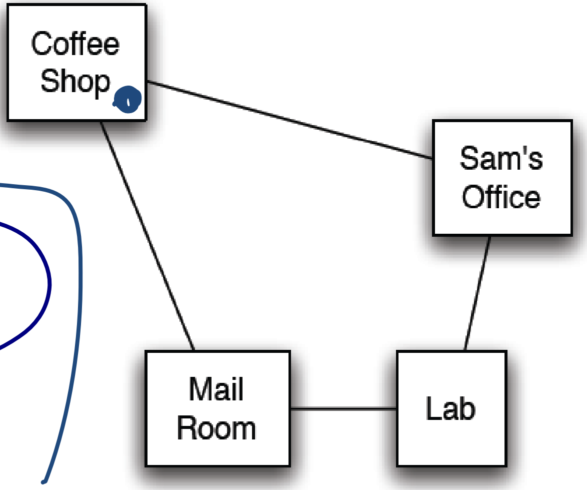
- A **plan** is a path from the state representing the initial state to a state that satisfies the goal.

Example state-space graph: first level

Goal: $\overline{swc}$

Start state





Actions

mc: move clockwise

mac: move anticlockwise

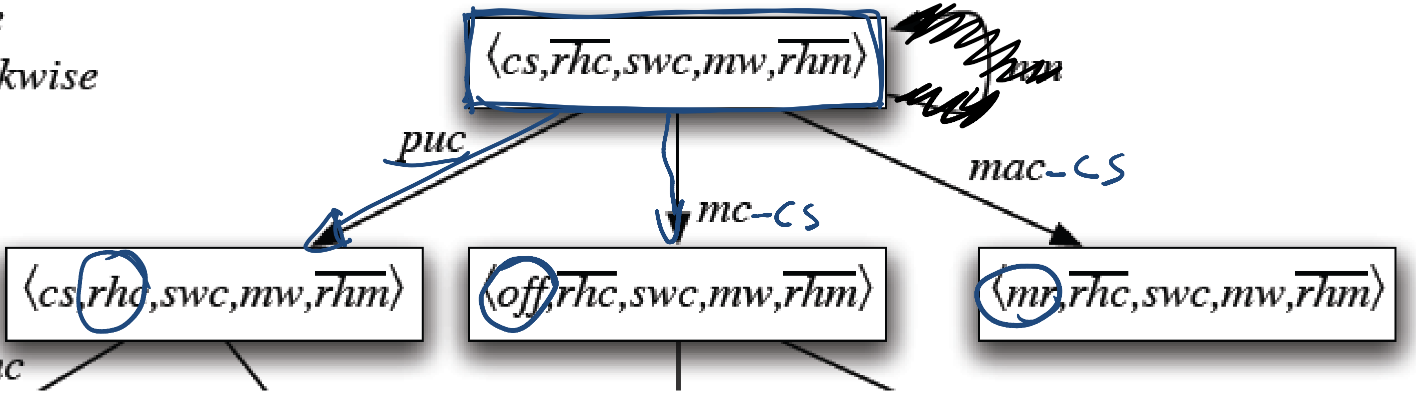
[redacted]

puc: pick up coffee

dc: deliver coffee

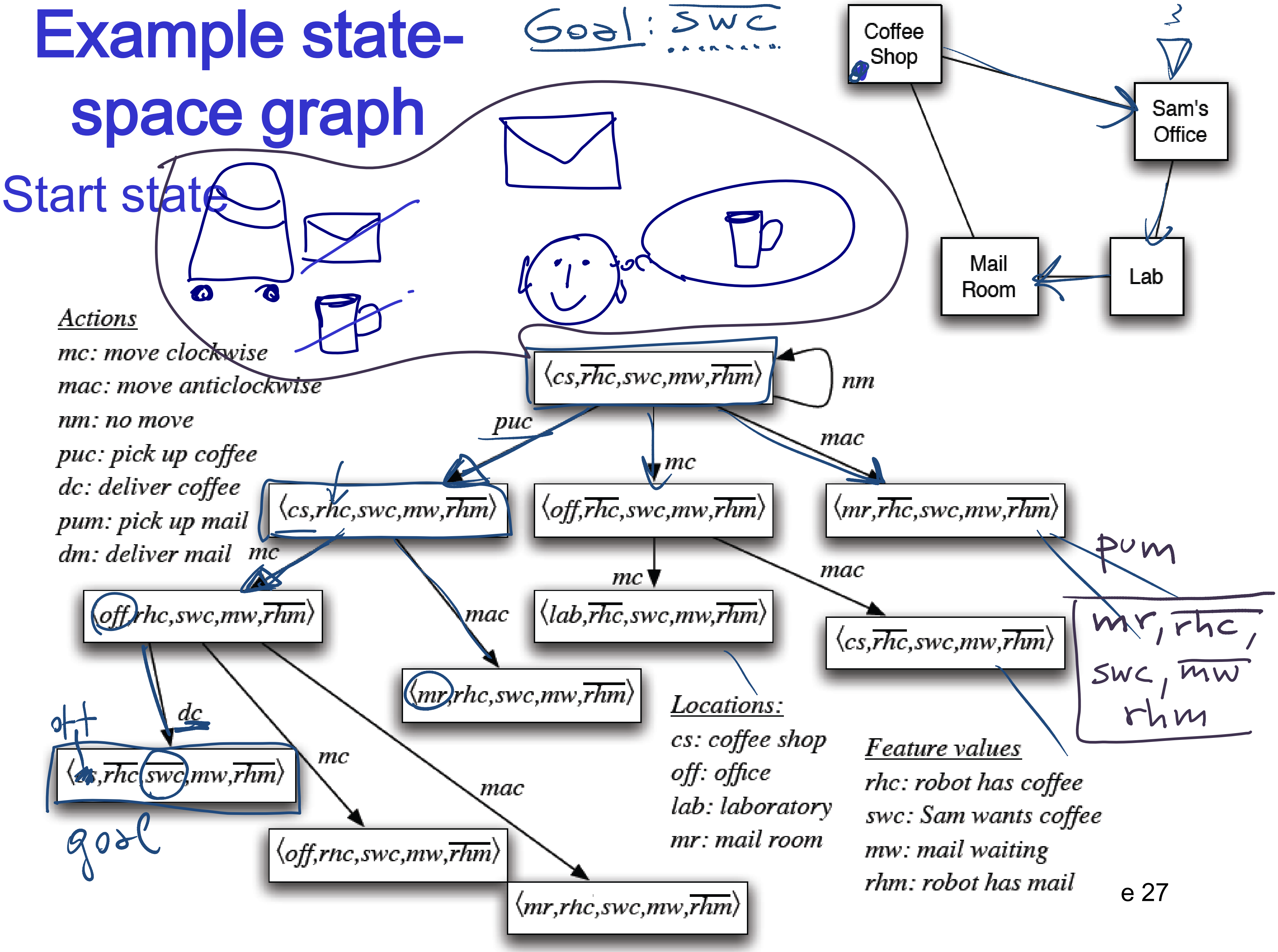
pum: pick up mail

dm: deliver mail



Example state-space graph

Goal: swc



Actions

mc: move clockwise
mac: move anticlockwise
nm: no move
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

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 valid actions
  - 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

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

In planning this is the…# actions………….

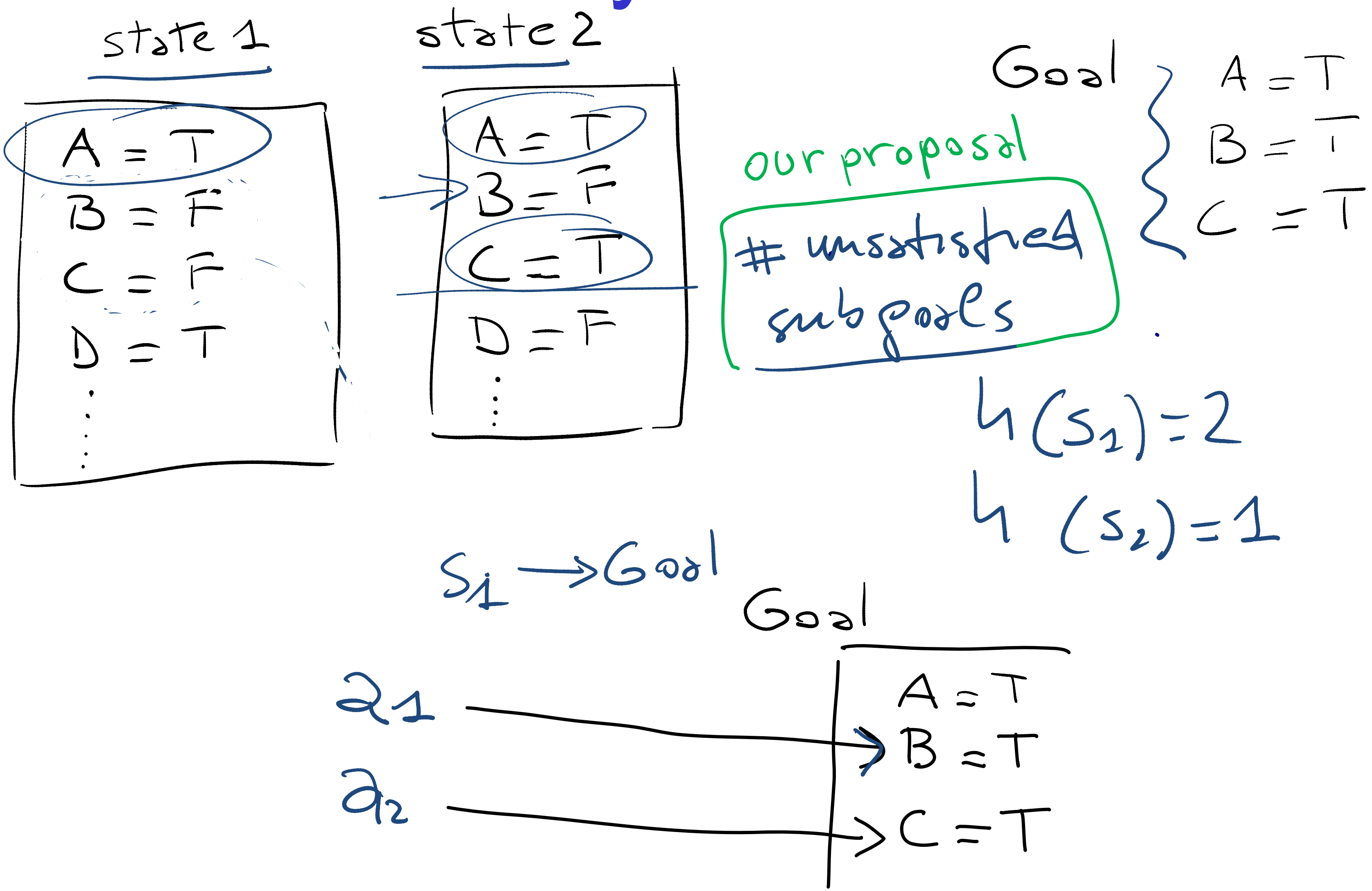
Two simplifications in the representation:

- 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?



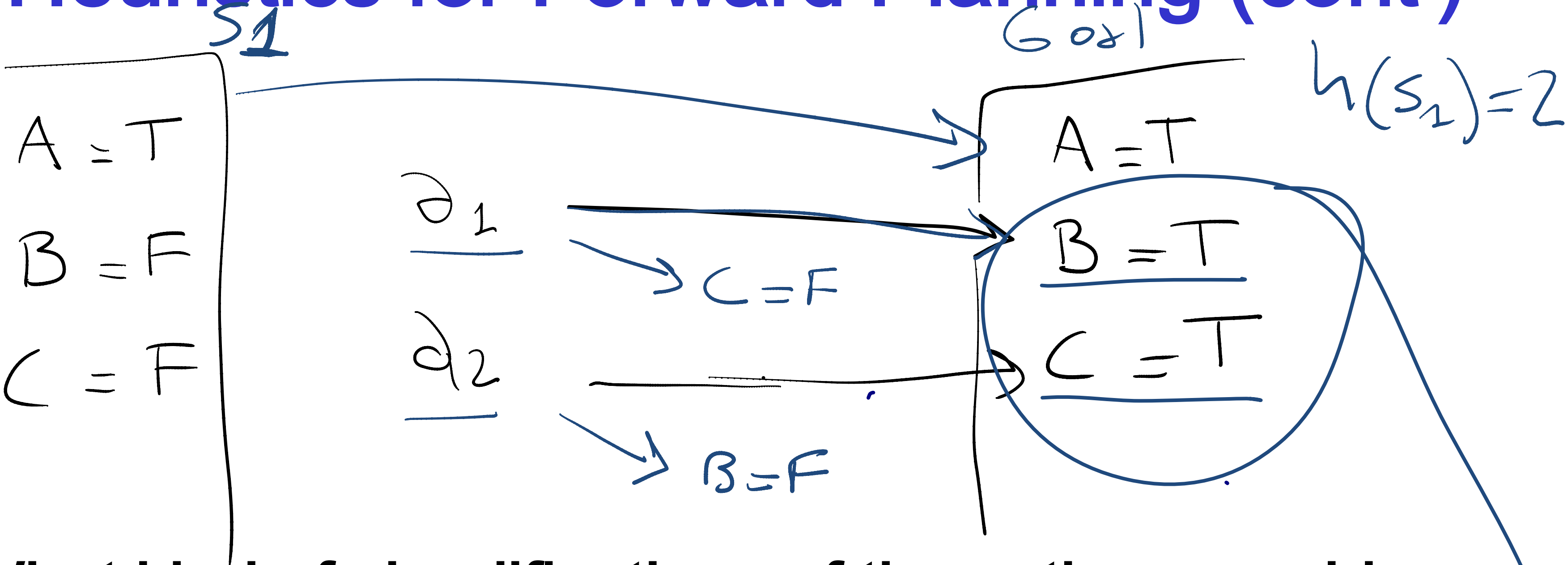
our proposal

unsatisfied subgoals

$h(s_1) = 2$

$h(s_2) = 1$

Heuristics for Forward Planning (cont')



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

too strong!

VERY STRONG

a) We have removed all ……………. preconditions

b) We have removed all ……………. "negative" effects

c) We assume no action can achieve…………………… both

INADMISSIBLE

Heuristics for Forward Planning: empty-delete-list

.30

- **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*)

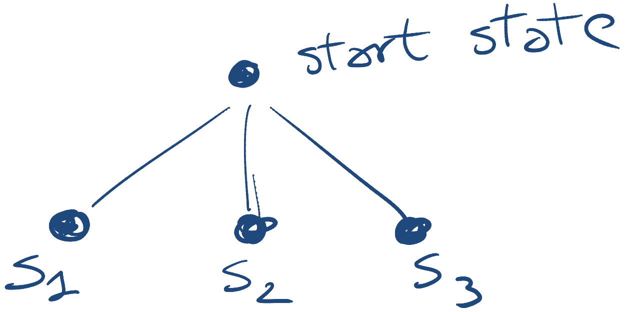
- **But then how do we compute the heuristic?**

……solve a simplified planning problem……

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



$h(s_1)$?

$h(s_2)$?

$h(s_3)$?

Let assume we are using A^*

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.

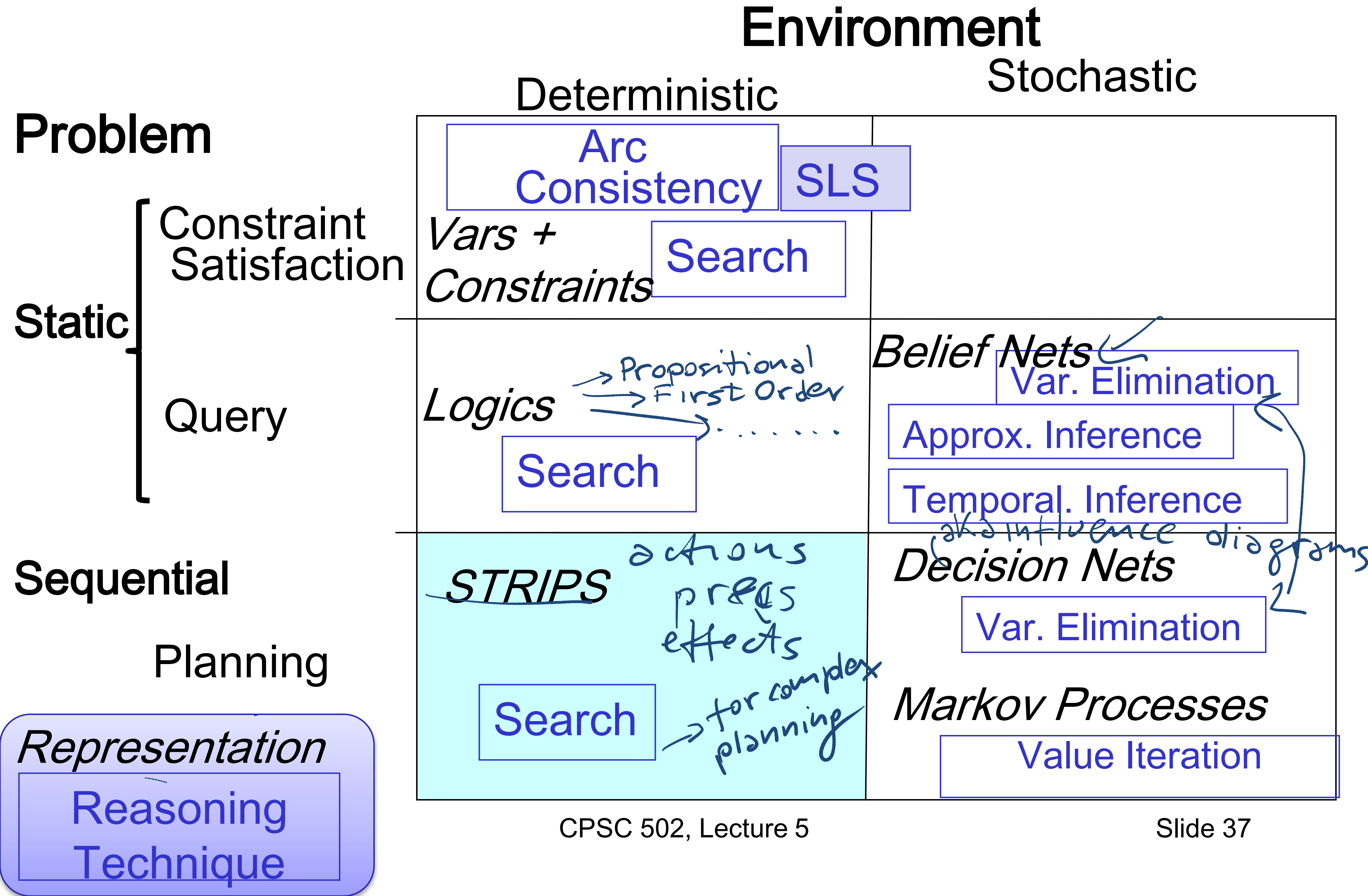
So to compute h for a given state we need to solve a planning problem (but a simpler one!)

You may need to do this MANY times

Today Sept 22

- Finish Stochastic Local Search (SLS)
- Planning
  - STRIPS – Forward Planning
  - Heuristics
  - **STRIPS -> CSP**

R&Rsys we'll cover in this course



Environment

Deterministic

Stochastic

Problem

Static

Constraint Satisfaction

Vars + Constraints

Arc Consistency

SLS

Search

Query

Logics

→ Propositional
→ First Order
→ . . . .

Search

Belief Nets

Var. Elimination

Approx. Inference

Temporal. Inference

Sequential

Planning

STRIPS

actions
precs
effects

Search

→ for complex planning

aka influence diagrams

Decision Nets

Var. Elimination

Markov Processes

Value Iteration

Representation

Reasoning Technique

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

a_1

a_2

A_0 B_0 C_0 0

a_{10} a_{20}

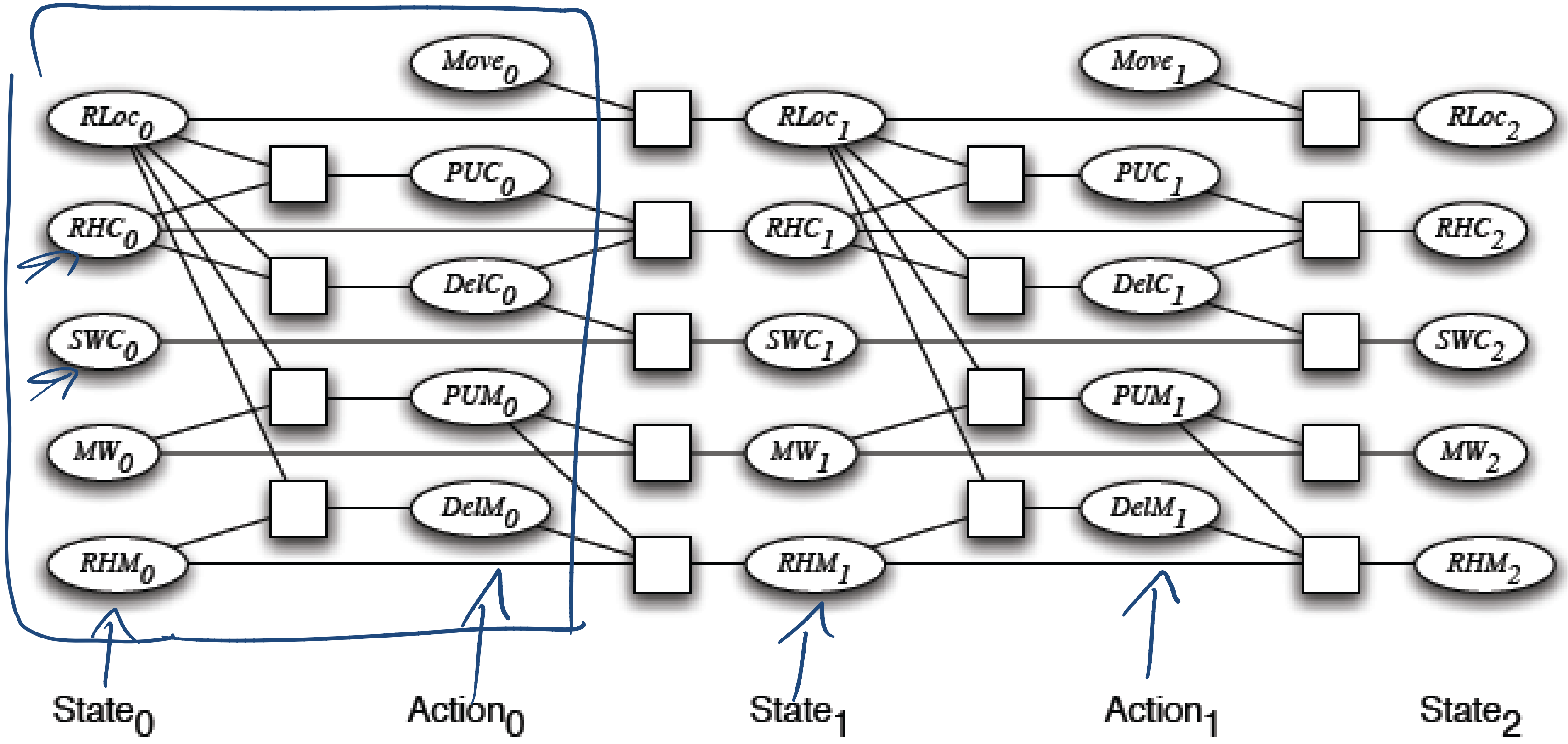
A_1 B_1 C_1 1

a_{11} a_{21}

A_2 B_2 C_2 2

a_{12} a_{22}

CSP Planning: Robot Example



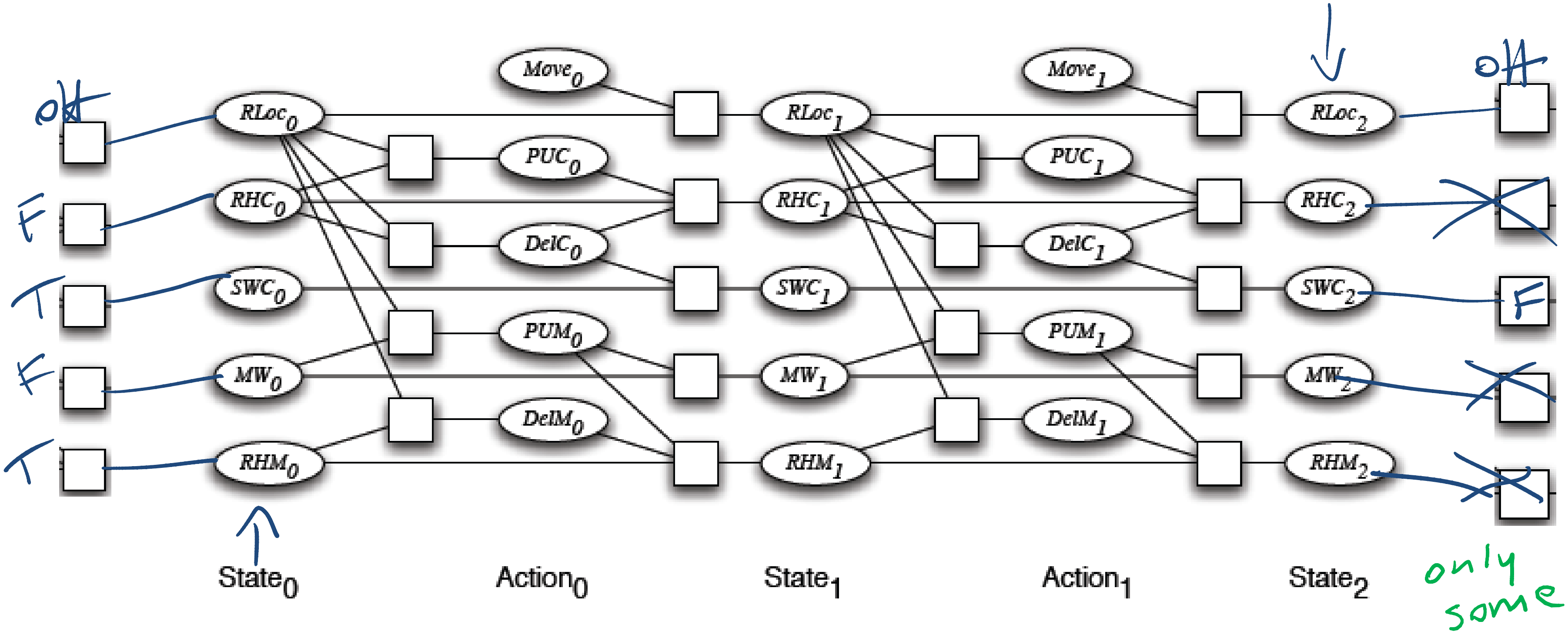
Variables for actions …. binary

action (non) occurring at that step

CSP Planning: Initial and Goal Constraints

unary

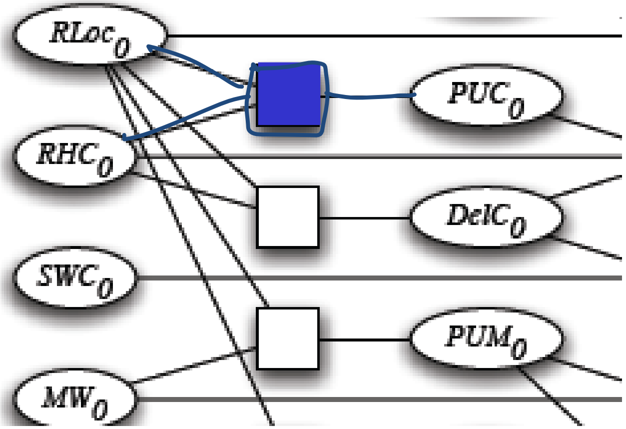
- initial state constraints constrain the state variables at time 0
- goal constraints constrain the state variables at time *k*



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



PUC_0

(pick up coffee)

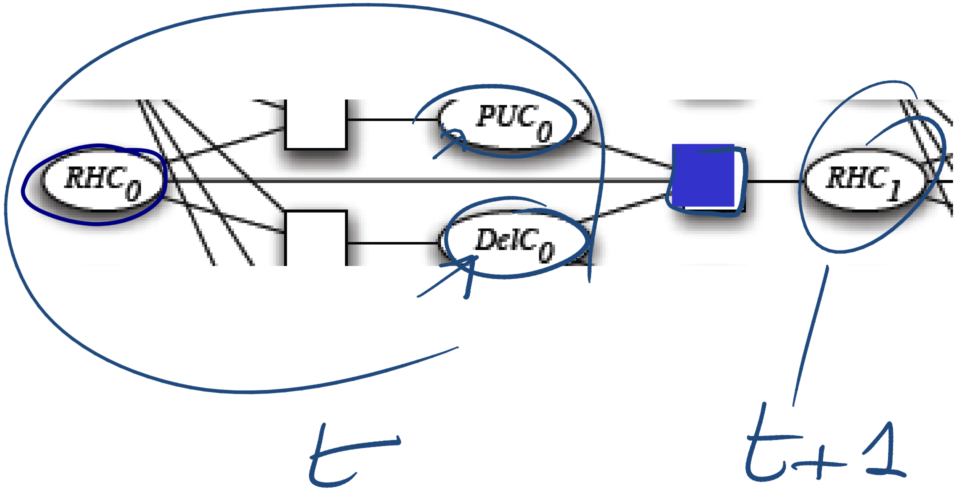
Rob Location

Rob has coffee

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

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	T
T	T	F	F
T	F	T	T
…	…	…	…
…	…	…	…

CSP Planning: Constraints Contd.

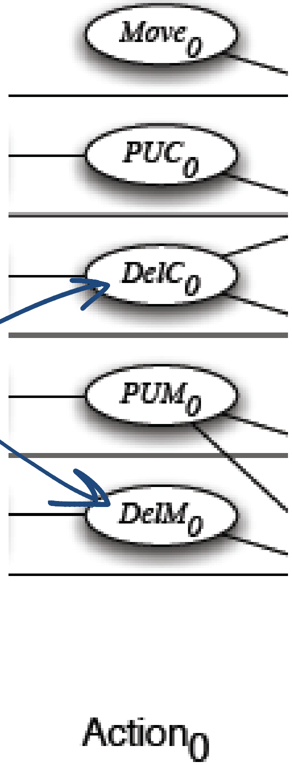
Other constraints we may want are action constraints:

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

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

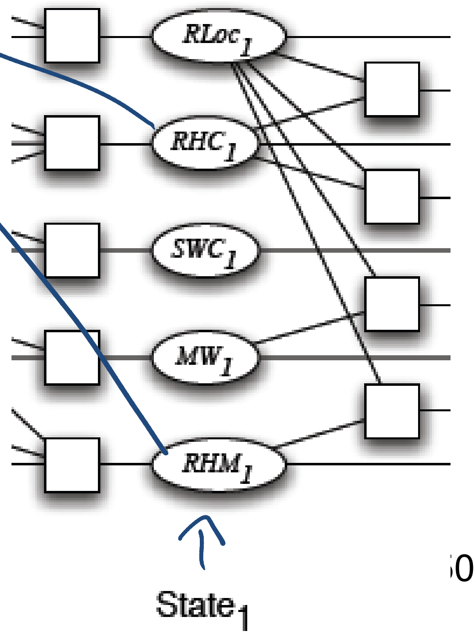


CSP Planning: Constraints Contd.

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

RHC_i	RHM_i
T	F
F	T
F	F



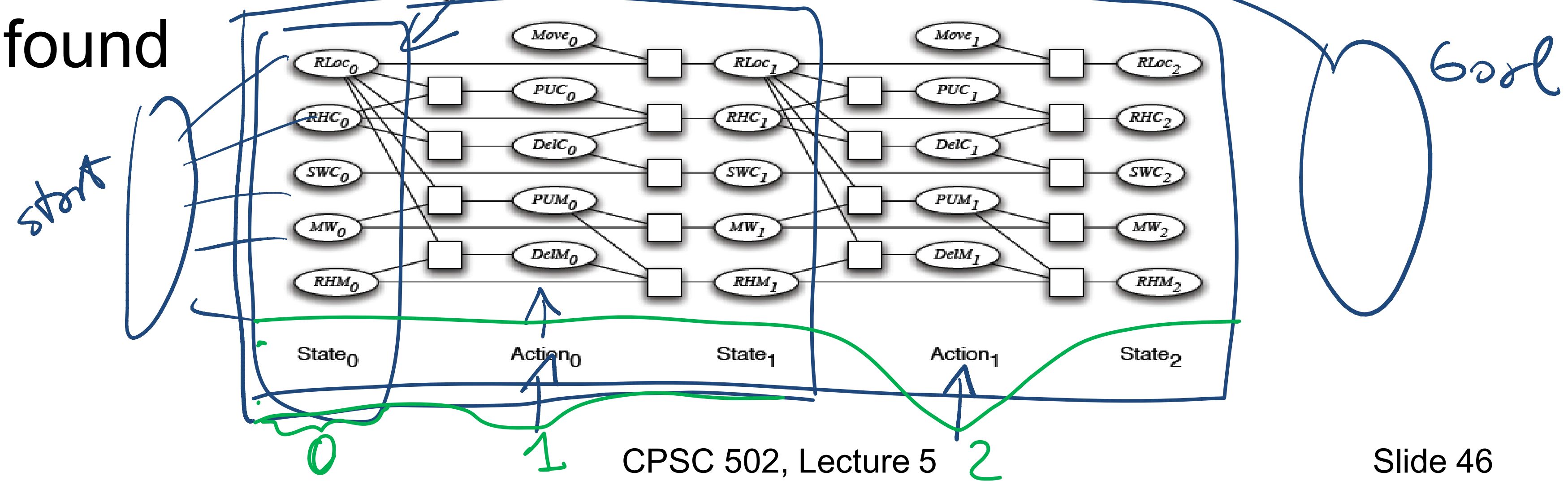
CSP Planning: Solving the problem

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

Run **arc consistency**, **search**, **stochastic local search**!

Plan: all actions with assignment T

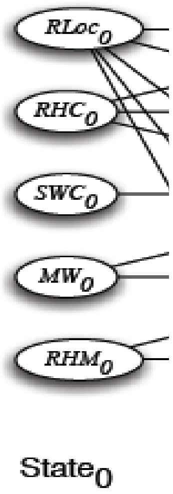
In order to find a plan, we expand our constraint network one layer at the time, until a solution is found



CSP Planning: Solving the problem

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

Run arc consistency, search, stochastic local search!



K = 0

Is $State_0$ a goal?

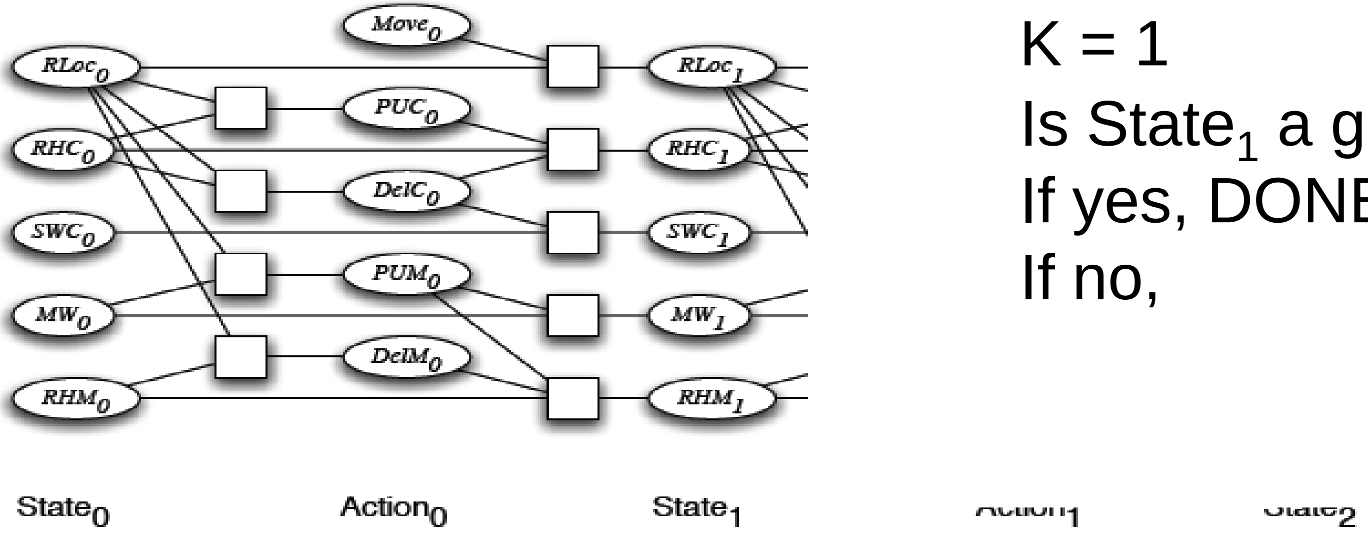
If yes, DONE!

If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon k =1

Run arc consistency, search, stochastic local search!



K = 1

Is State$_1$ a goal

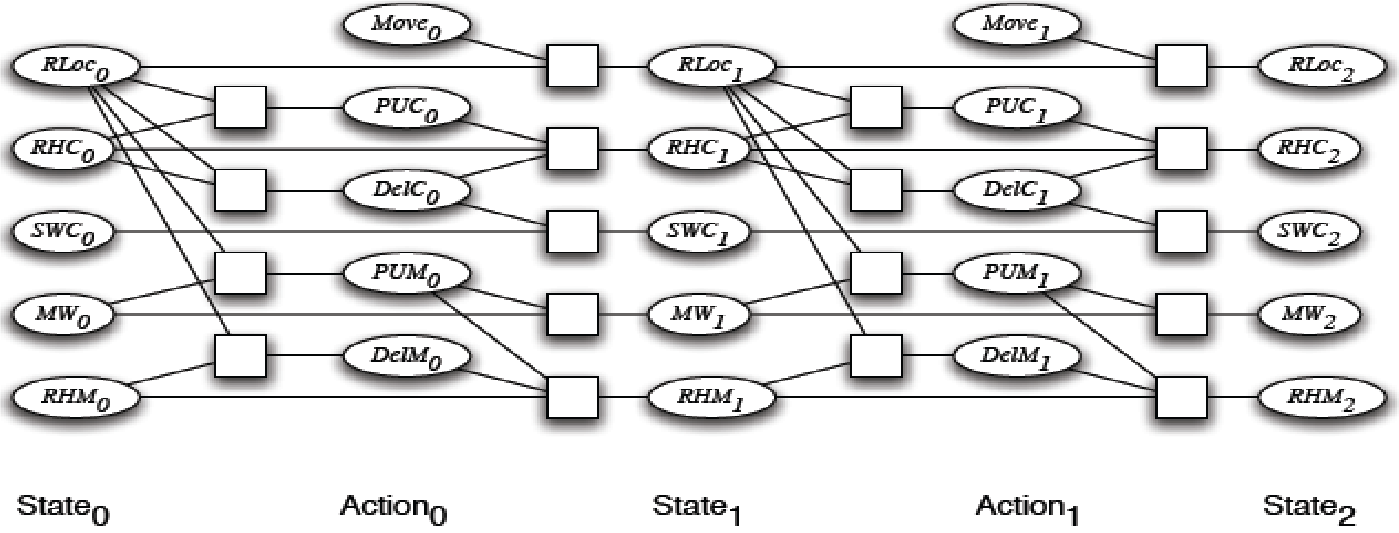
If yes, DONE!

If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon k = 2

Run arc consistency, search, stochastic local search!



K = 2: Is State$_2$ a goal

If yes, DONE!

If no….continue

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

- In general, Planning graphs are an efficient way to create a representation of a planning problem that can be used to
    - Achieve better heuristic estimates
    - Directly construct plans

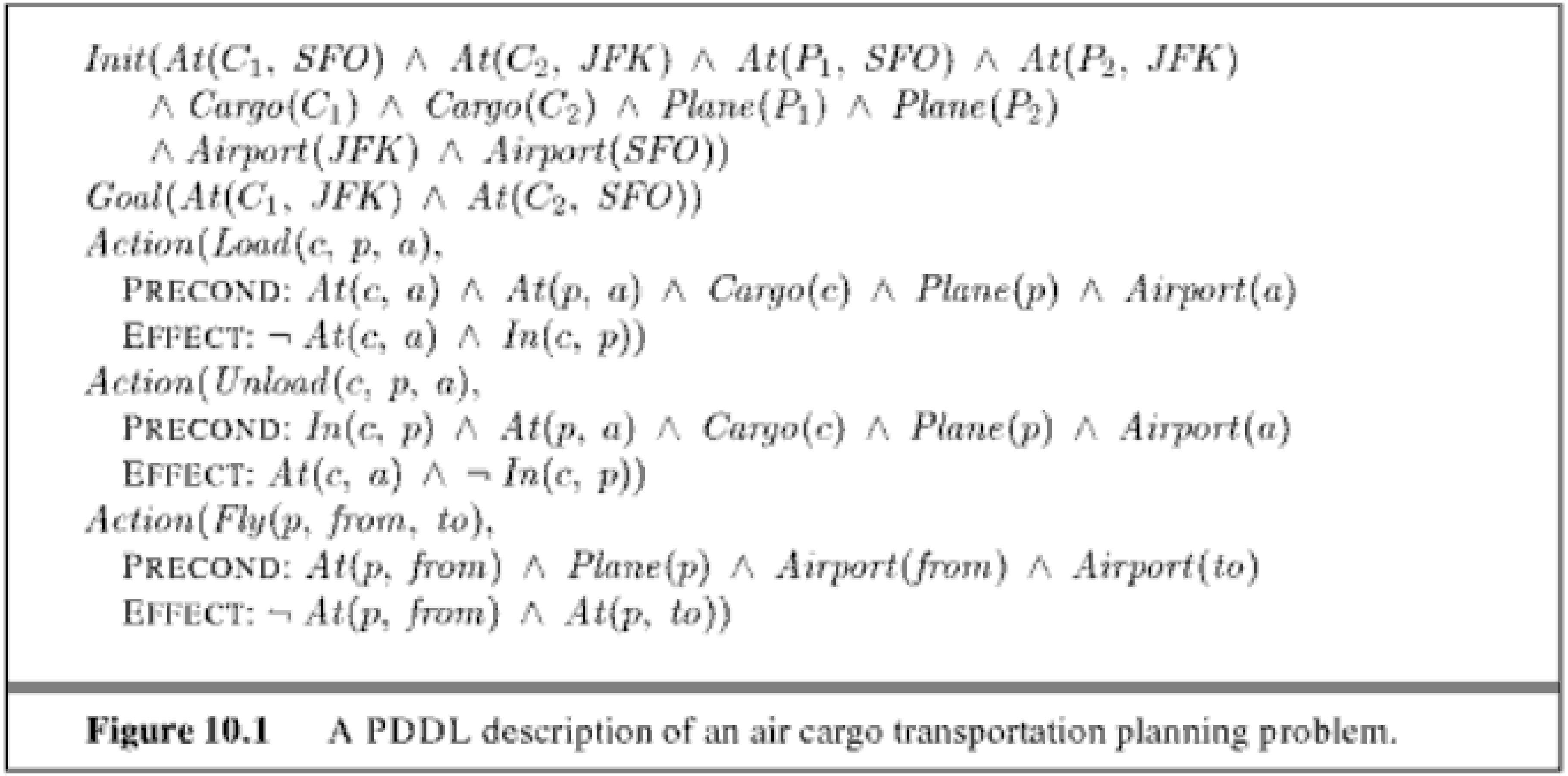
TODO for next Tue

Read Chp 5 of textbook (Logics) up to 5.3.3 included

- Do exercise 8.A, B, C
  http://www.aispace.org/exercises.shtml
  Please, look at solutions only after you have tried hard to solve them!

Start working on assignment-1 !

Expressiveness of the language



$Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)$
$\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)$
$\quad \land Airport(JFK) \land Airport(SFO))$
$Goal(At(C_1, JFK) \land At(C_2, SFO))$
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $\neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
EFFECT: $\neg At(p, from) \land At(p, to))$

Figure 10.1 A PDDL description of an air cargo transportation planning problem.

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

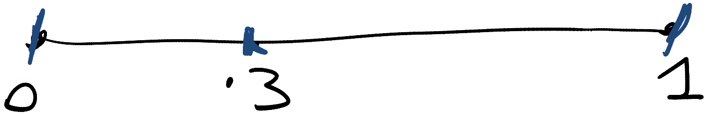
Practice exercise using STRIPS to CSP will be posted next week (maybe a couple ☺)

Sampling a discrete probability distribution

e.g. Sim. Annealing. Select n' with probability p

p = .3

generate random number in [0,1]



If < .3 accept n'

e.g. Beam Search: Select K individuals. Probability of selection proportional to their value

SAME HERE

→ n_1 $p_1 = .1$

→ n_2 $p_2 = .3$

→ n_3 $p_3 = .2$

→ n_4 $p_4 = .4$

n_3 first sample

n_1 second sample

⋮