Planning + Intro Logics

Computer Science cpsc322, Lecture 6

(Textbook Chpt 8.1 -8.2, 8.4)

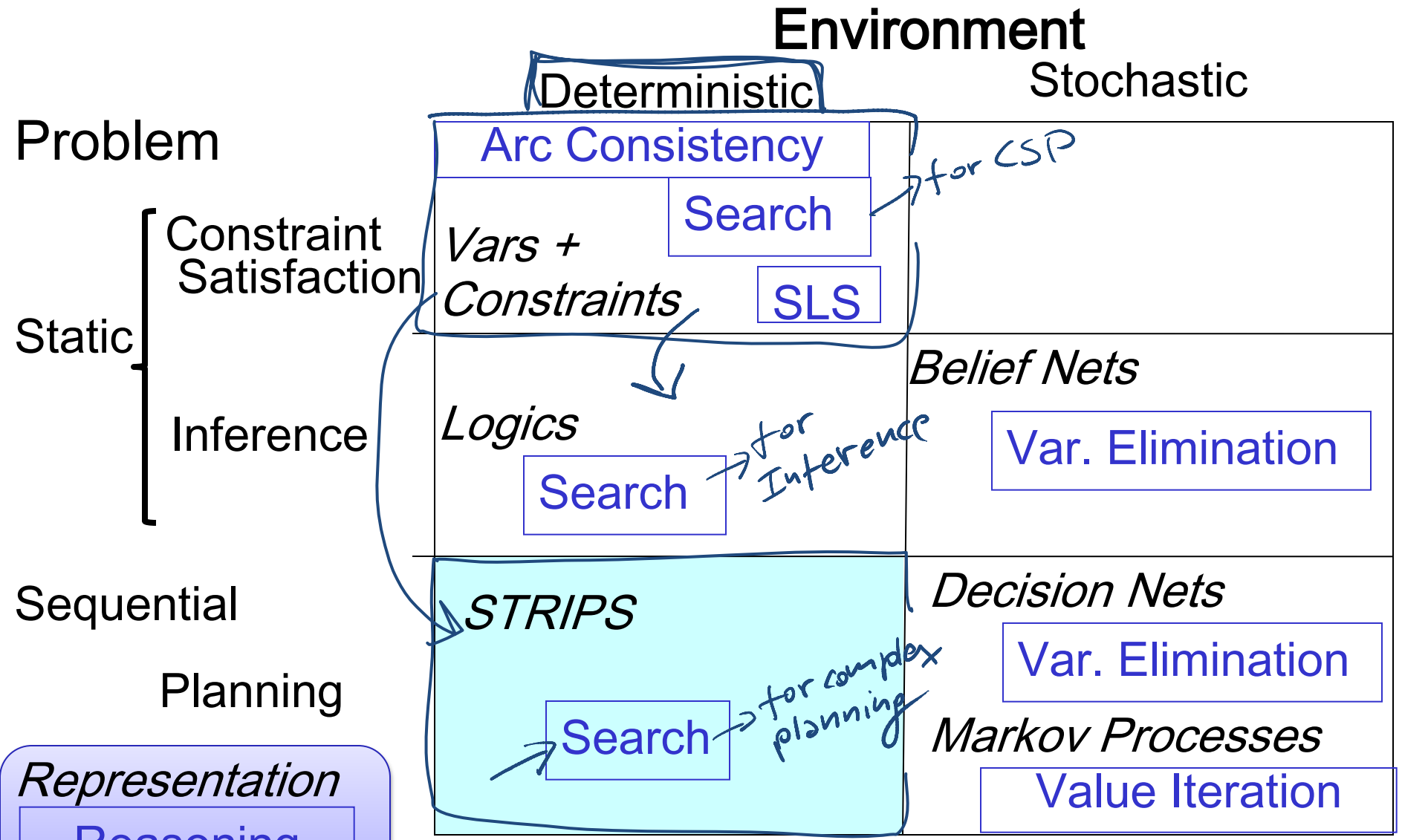
May, 23, 2012

Course Announcements

- Assignment2 on CSPs due now !

- **MIDTERM: Mon May 28th – 3PM (room 310)**
- **How to prepare: end of this lecture**

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



Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

- **Logic Intro**
  - Propositional Definite Clause Logic: Syntax

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 actions 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)

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

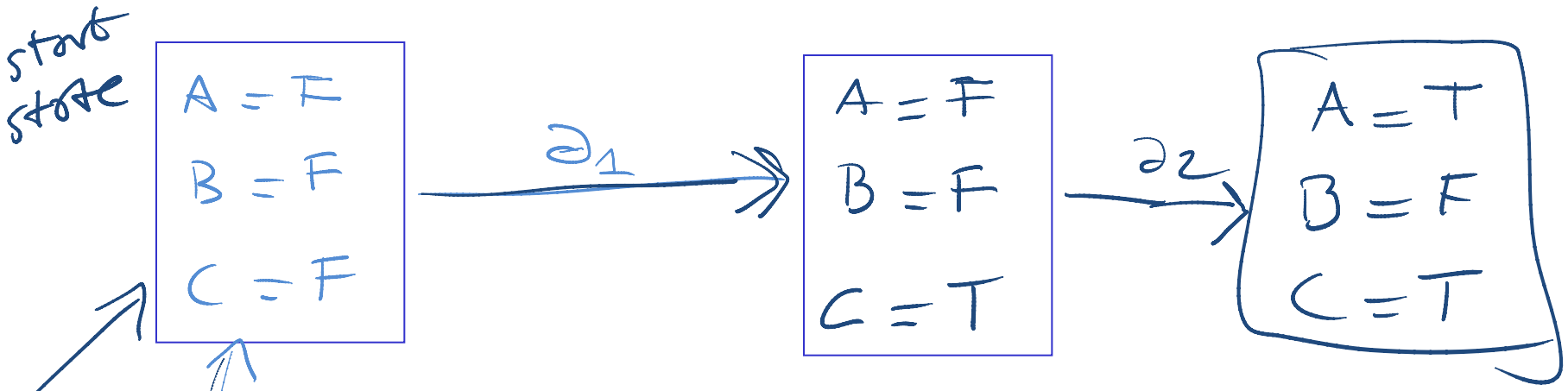
Goal: Agent wants to be in a possible world were **some** variables are given specific values

sample goal

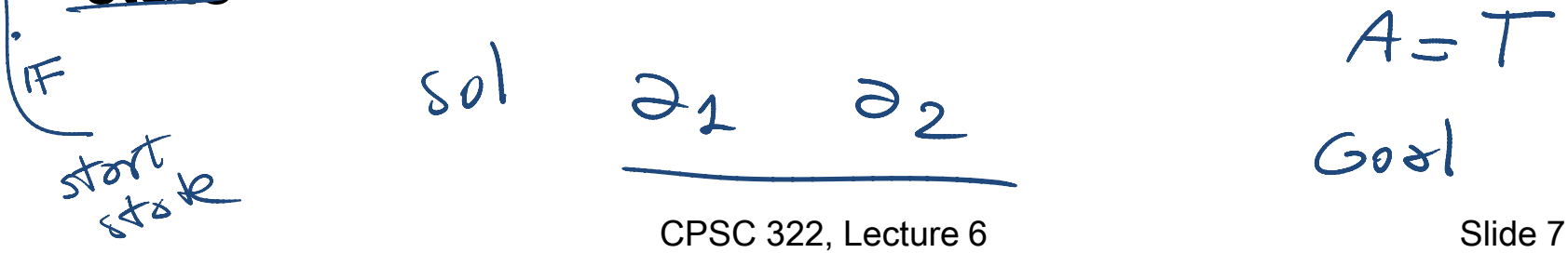
$\begin{bmatrix} A=T & C=F \end{bmatrix}$

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



Lecture Overview

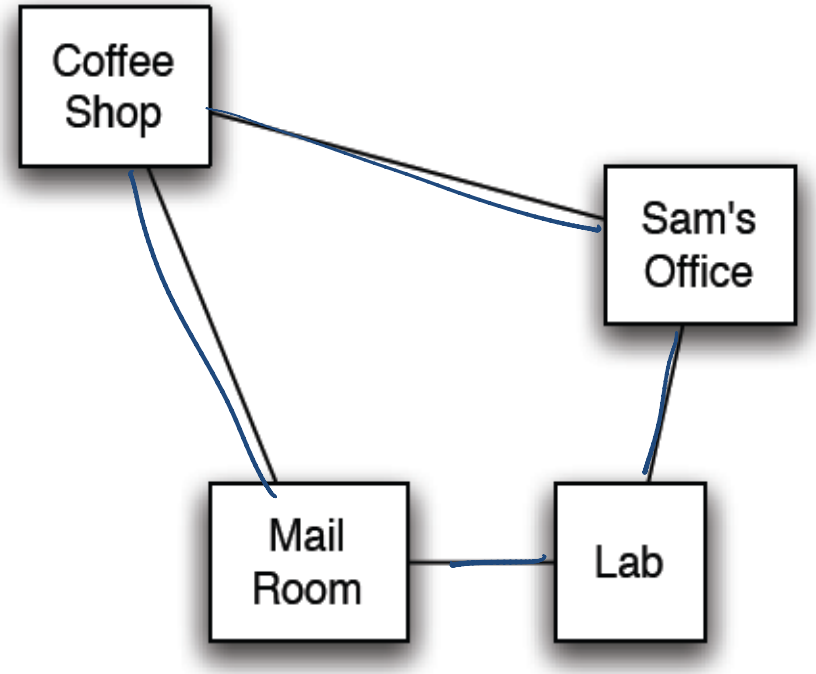
Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

- **Logic Intro**
  - Propositional Definite Clause Logic: Syntax

Delivery Robot Example (textbook)

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



Another example is available as a Practice Exercise: "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 - Rob has coffee True/False.

SWC - Sam wants coffee T/F

MW - Mail is waiting T/F

RHM - Rob has mail T/F

two different notations

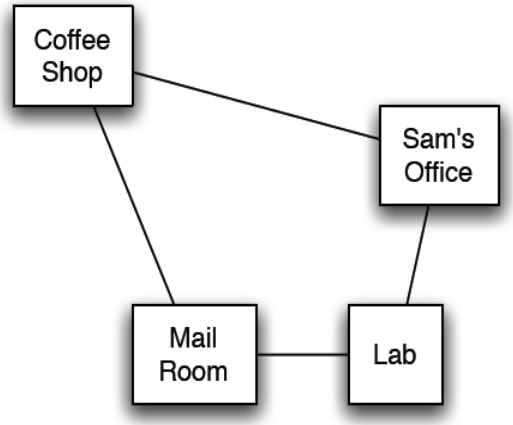
4 2 2 2 2

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

Number of states: 64

32 64 48 16

Delivery Robot Example: Actions



The robot's actions are:

Move - Rob's move action

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

~~not move (stay)~~

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

Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

- **Logic Intro**
  - Propositional Definite Clause Logic: Syntax

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: Example

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* :

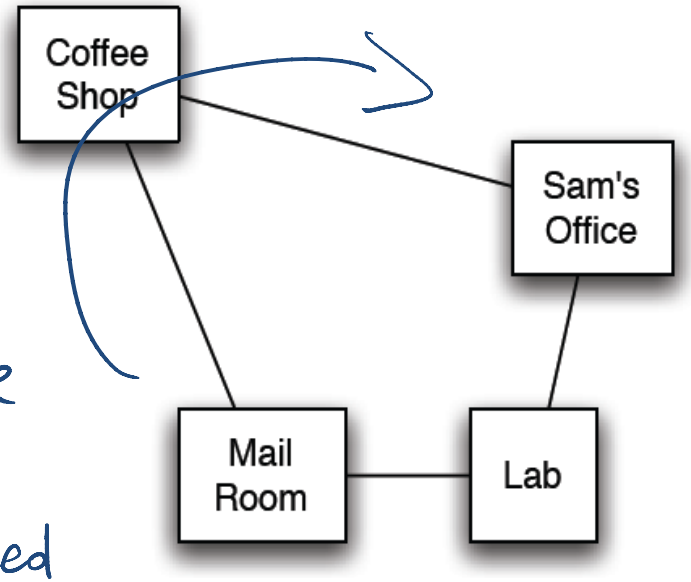
- **preconditions** *Loc* = *off* and *RHC* = T (SWC = T)
- **effects** RHC = F and SWC = F

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 ?



mc-cs

Prec loc=cs

Eff loc=office

mc-off

:

:

mac-cs

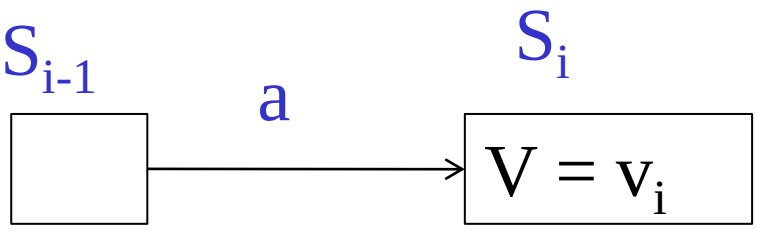
→ you need 4 specific actions

→ and 4 specific actions for move anticlockwise

STRIPS Actions (cont')

The STRIPS assumption: all features 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**?



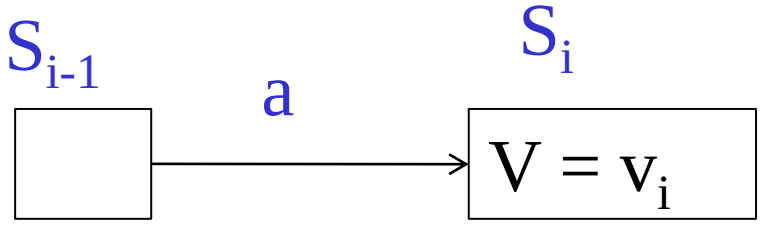
what can we conclude about *a* and/or the state of the world S_{i-1} ,immediately preceding the execution of *a*?

1 $V = v_i$ was TRUE in S_{i-1}

2 One of the effects of **a** is to set $V = v_i$

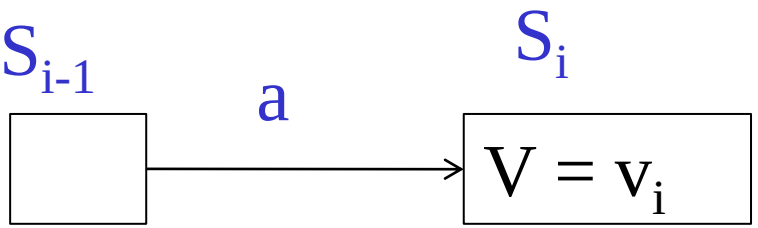
3 At least one of the above

4 None of the above



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

3 At least one of the above



Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

• Logic Intro

- Propositional Definite Clause Logic: Syntax

Forward Planning

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

- 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.

What actions *a* are possible in a state s?

Those where **a**'s effects are satisfied in **s**

Those where the state **s'** reached via **a** is on the way to the goal

Those where **a**'s preconditions are satisfied in **s**

Forward Planning

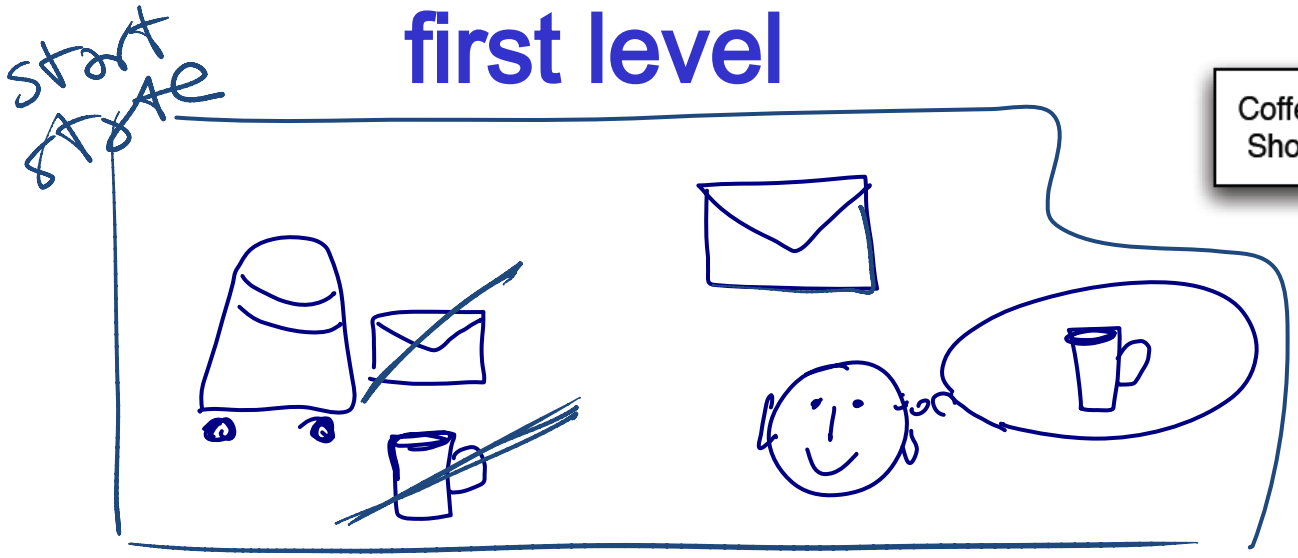
To find a plan, a solution: search in the state-space graph.

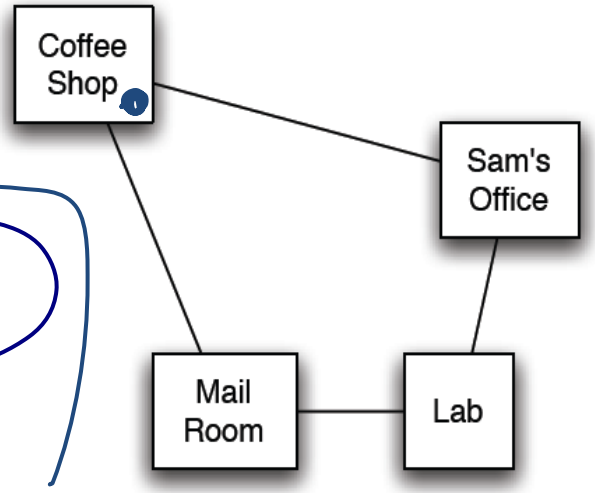
- The **states** are the **possible worlds**
- The **arcs** from a state ***s*** represent all of the **actions** that are legal in state ***s***.
- A **plan** is a path from the state representing the initial state to a state that satisfies the goal.

What actions *a* are possible in a state s?

Those where a's preconditions are satisfied in s

Example state-space graph: first level





Actions

mc: move clockwise

mac: move anticlockwise

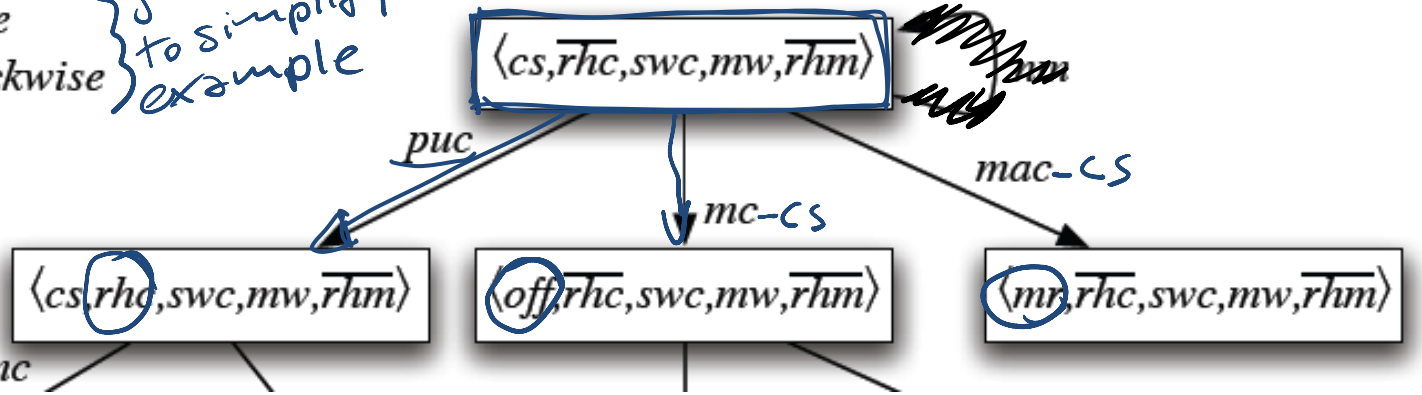
just two to simplify the example

puc: pick up coffee

dc: deliver coffee

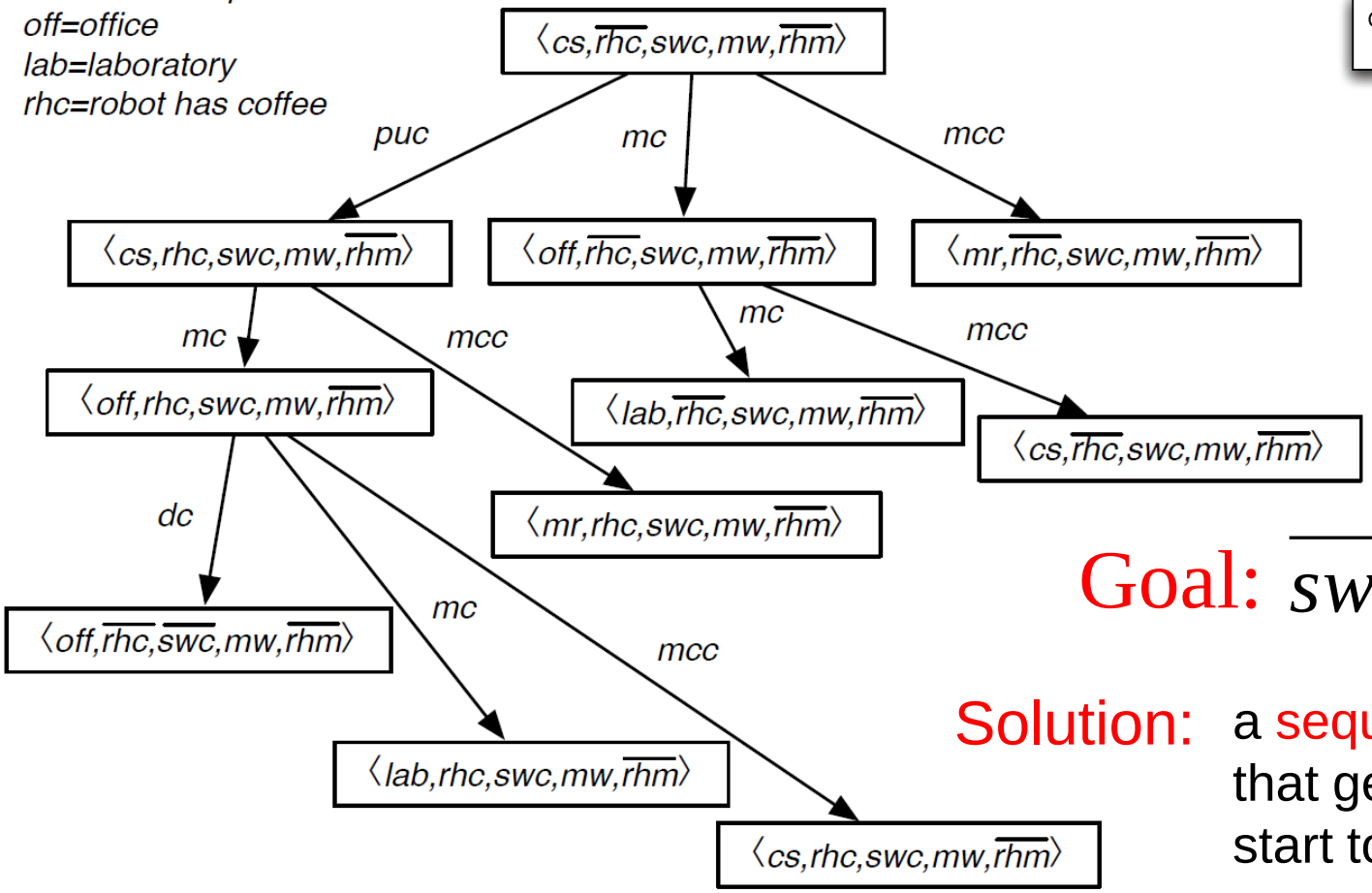
pum: pick up mail

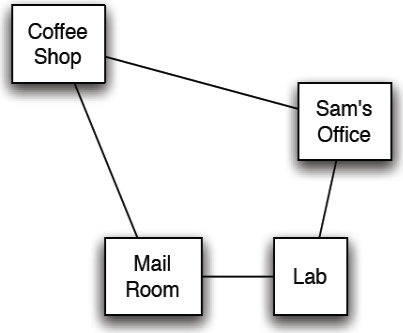
dm: deliver mail



Example for state space graph

cs=coffee shop
off=office
lab=laboratory
rhc=robot has coffee





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?

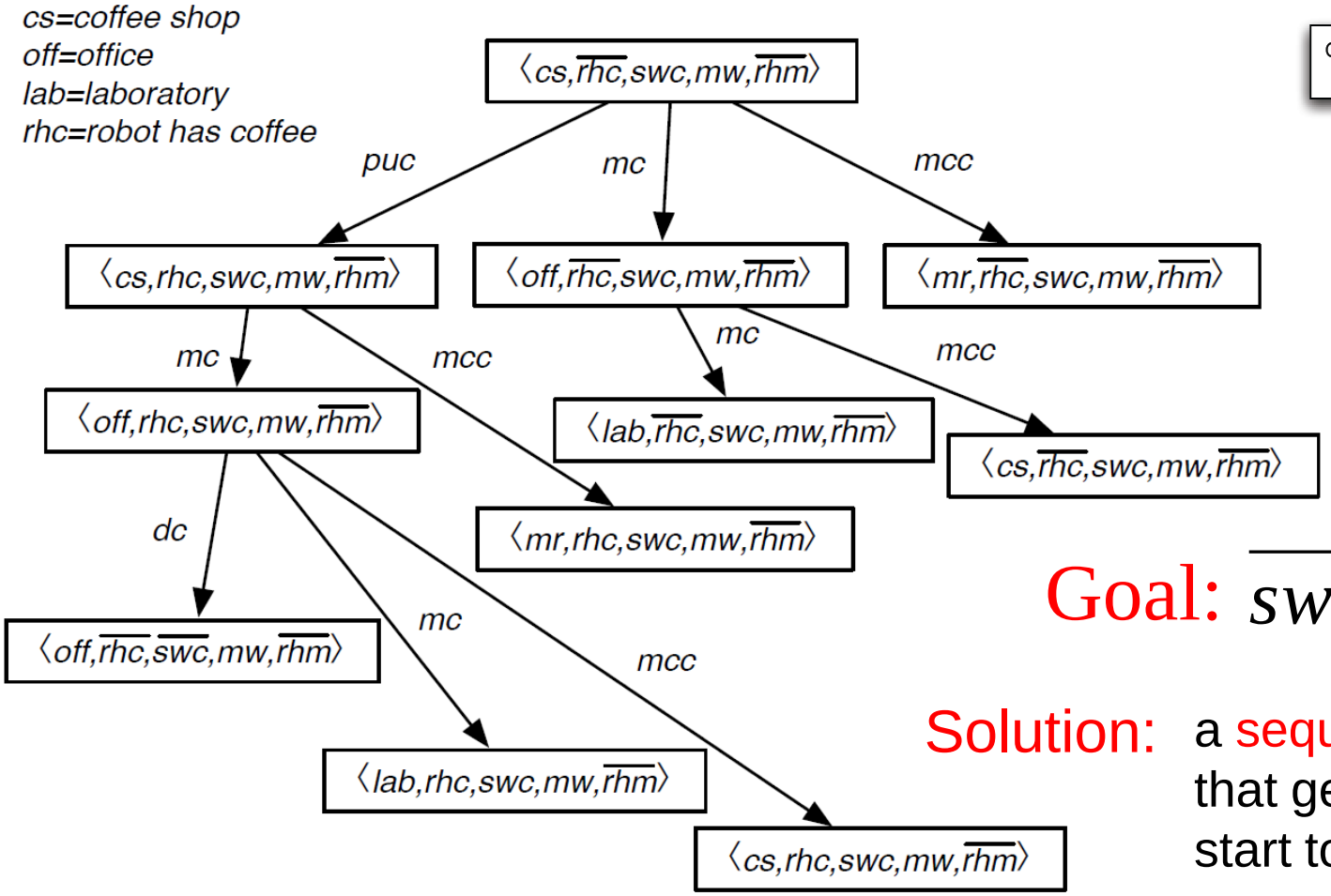
(puc, mc)

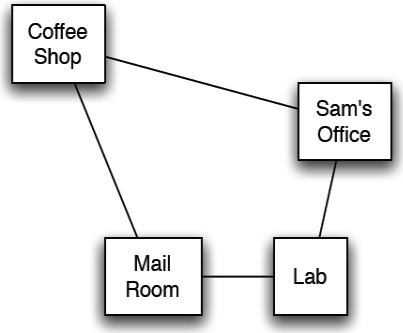
(puc, mc, mc)

(puc, dc)

(puc, mc, dc)

Example for state space graph





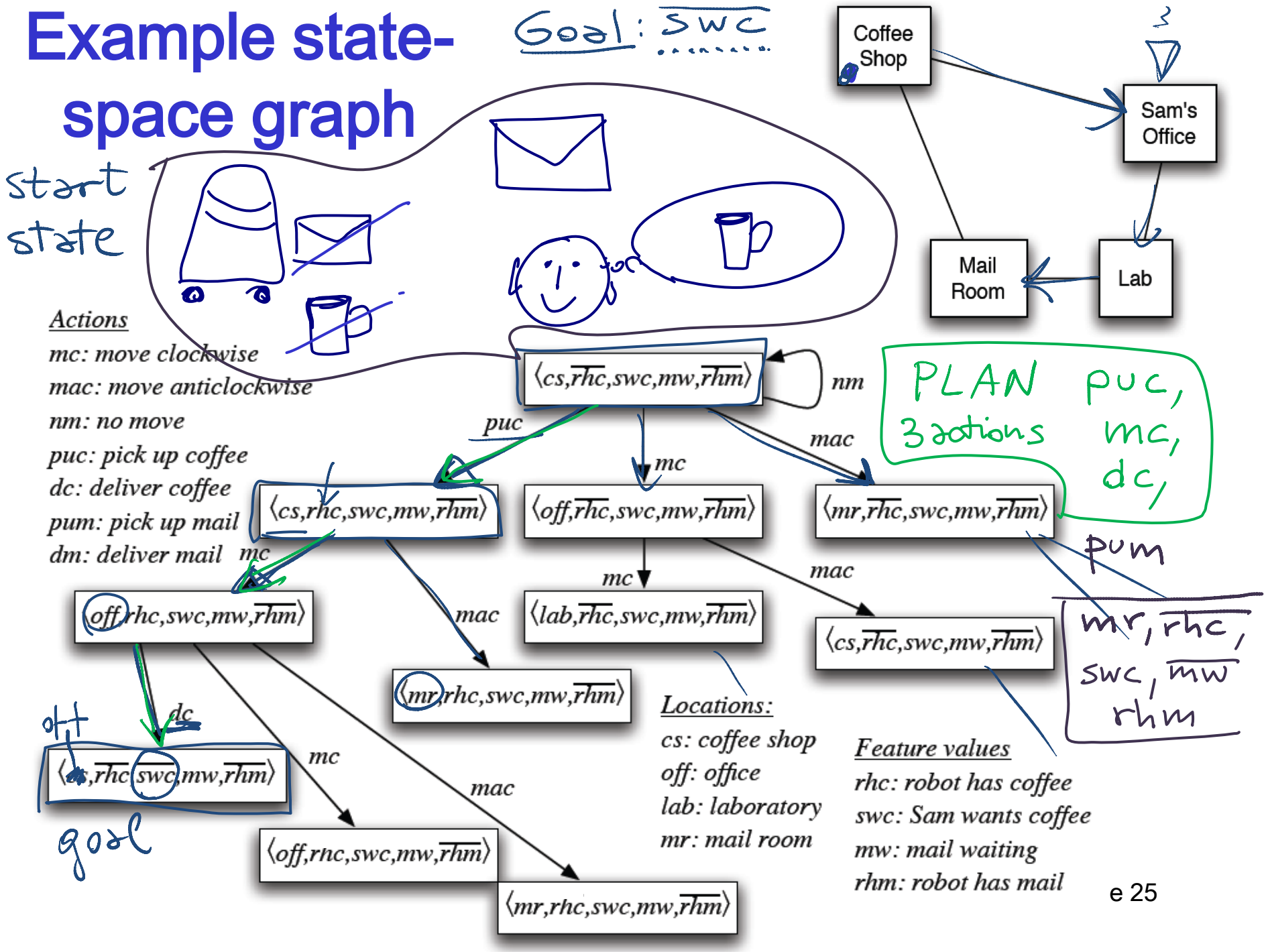
Goal: $\overline{swc}$

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

What is a solution to this planning problem?

(puc, mc, dc)

Example state-space graph



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 p. world full assign.
- Successor function states reachable by applying valid actions
- Goal test partial assign.
- Solution sequence of actions
- Heuristic function

Inference

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

Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- **Heuristics** (*not on textbook*)
- CSP Planning

• Logic Intro

- Propositional Definite Clause Logic: Syntax

Heuristics for Forward Planning

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

In planning this is the…#. actions……

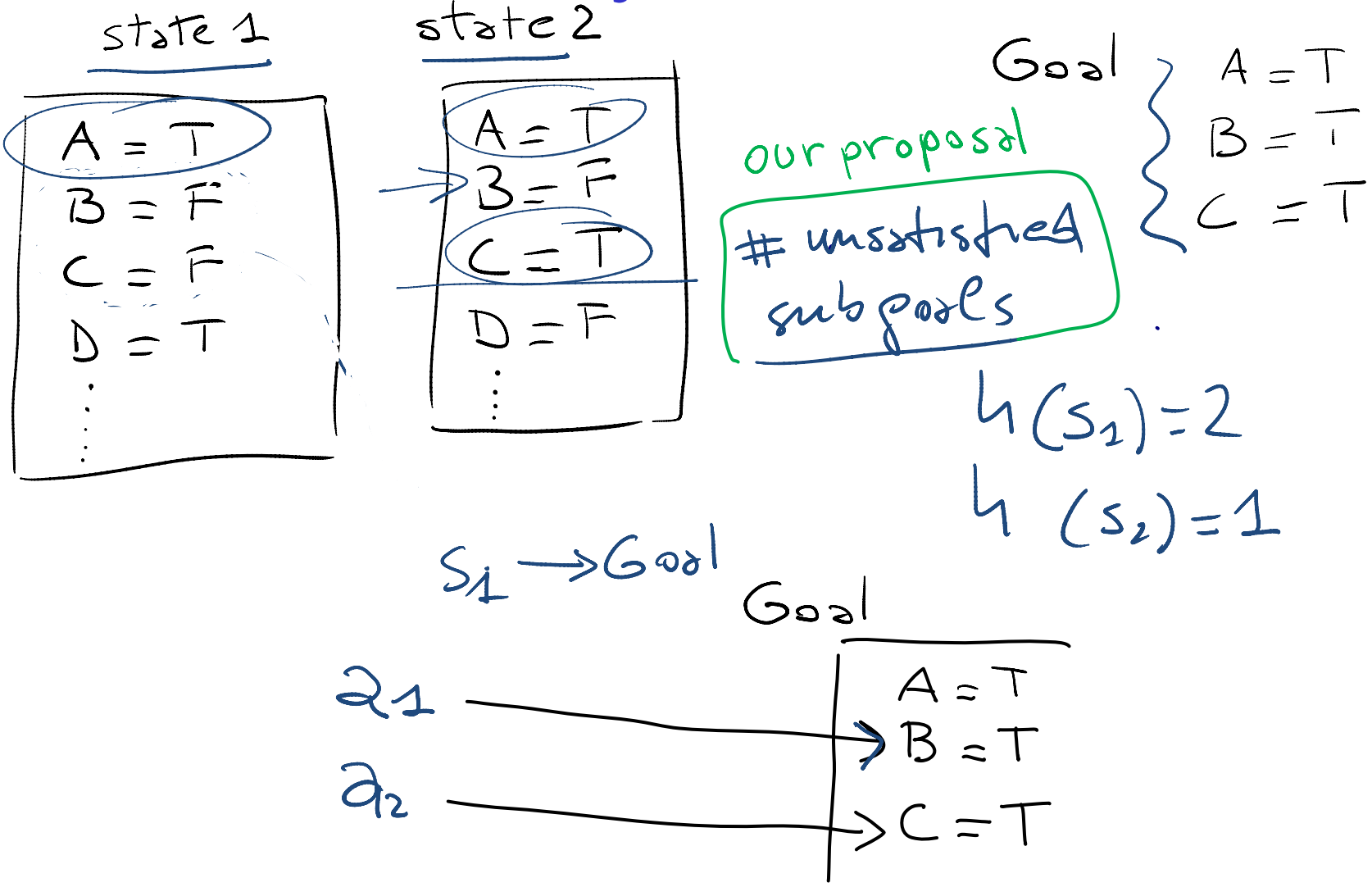
Two simplifications in the representation:

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

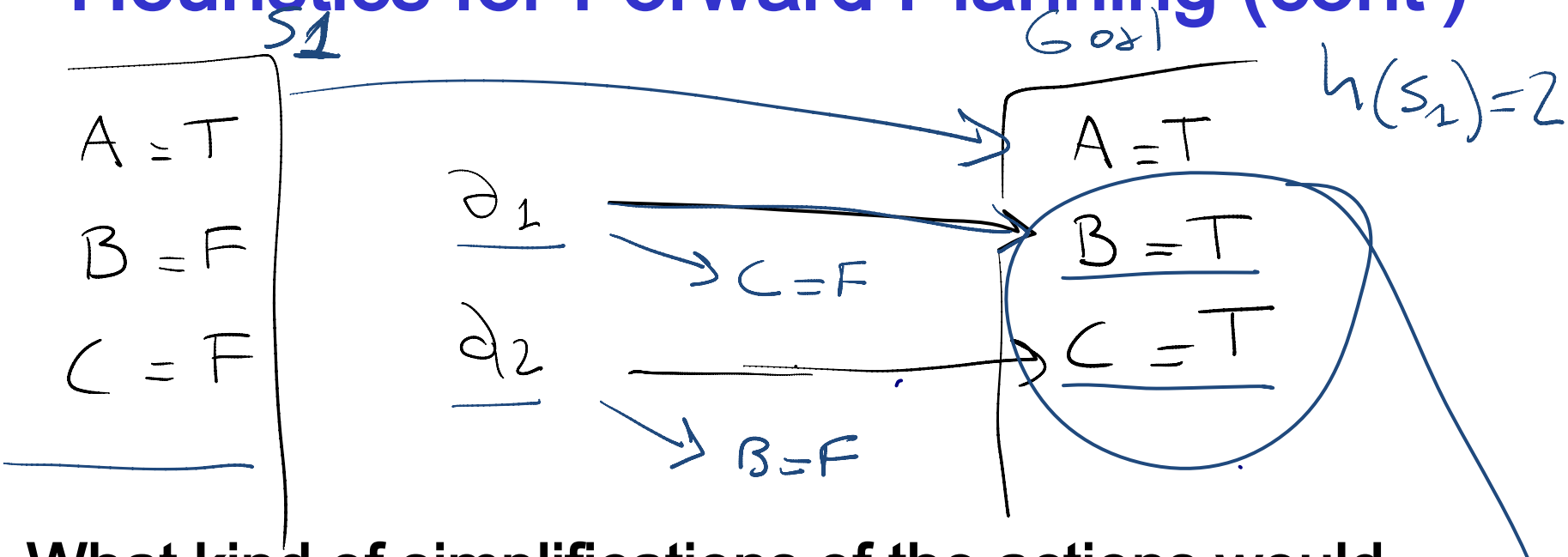
And a Def. a subgoal is a particular assignment in the goal e.g., if the goal is <A=T, B=T, C=T> then….

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

Heuristics for Forward Planning: Any ideas?



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

- **We only relax the problem according to (……)**

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

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

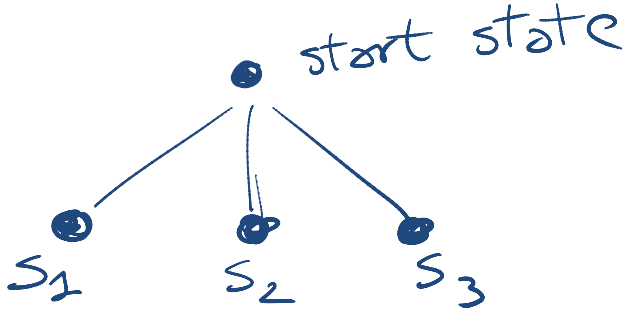
- **But then how do we compute the heuristic?**

……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



$h(s_1)$

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

Final Comment

- You should view Forward Planning as one of the basic planning techniques (we'll see another one after the break)
- 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 2002 and 2008 planning competition (posted in the class schedule)

Learning Goals for today's class – part-1

You can:

- 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.
- Construct and justify a **heuristic function** for forward planning.

Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning (Chp 8.4)

→ 30 MINS BREAK

- **Logic Intro**
  - Propositional Definite Clause Logic: Syntax

Planning as a CSP

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

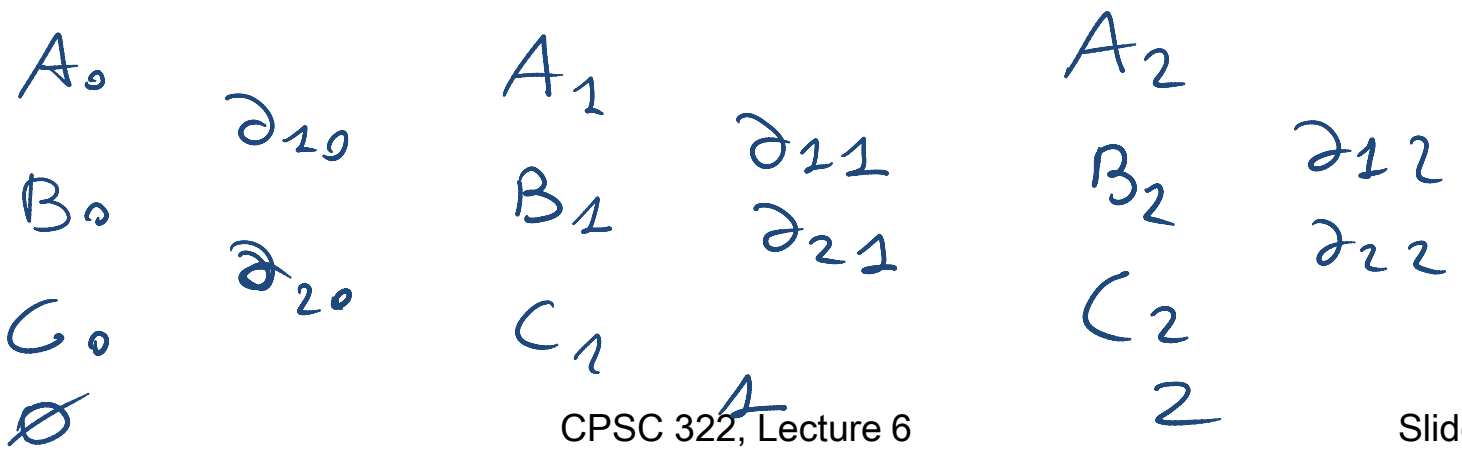
- We simply reformulate a STRIPS model as a set of variables and constraints

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

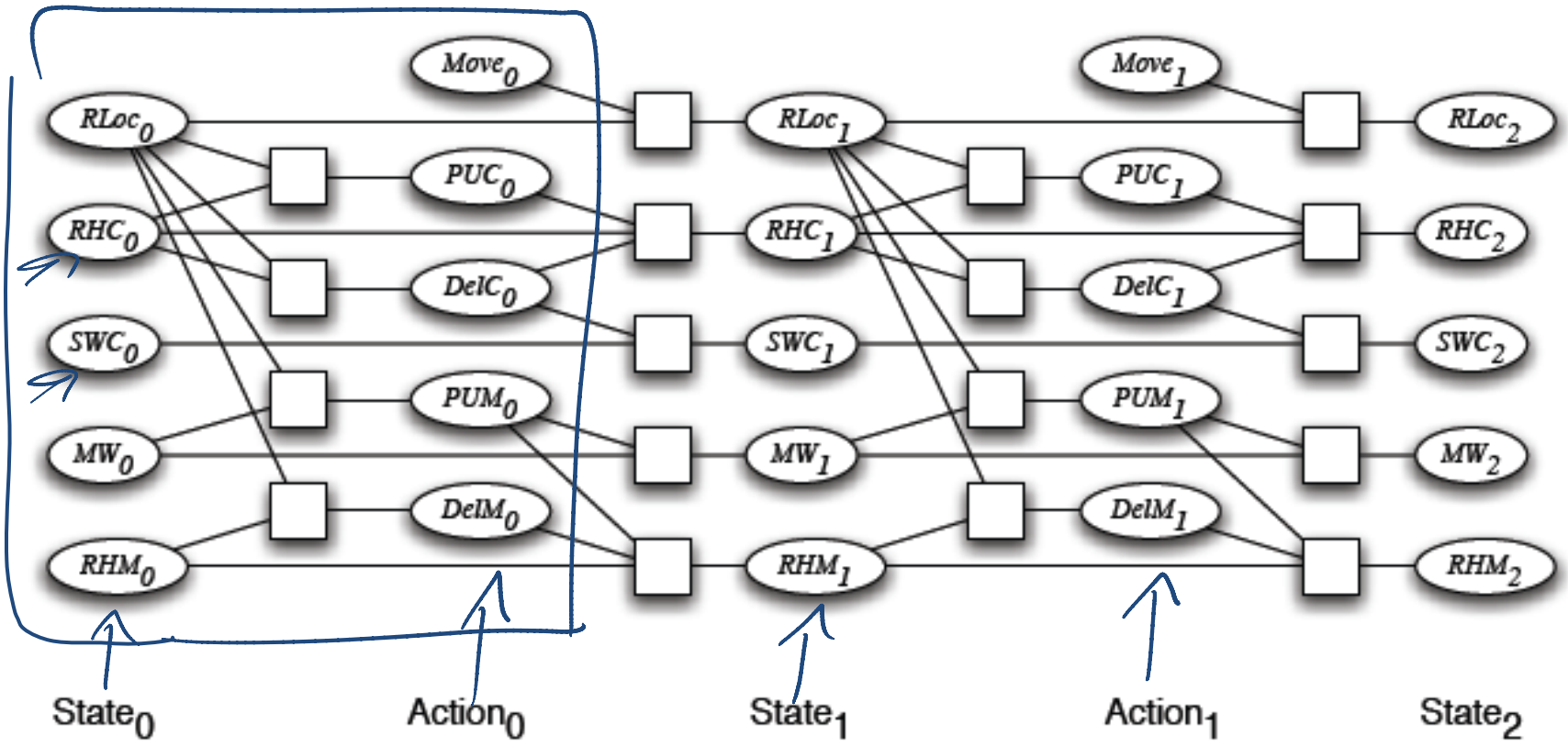
e.g., see Practice Exercise UBC commuting "*careAboutEnvironment*" constraint

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.



CSP Planning: Robot Example



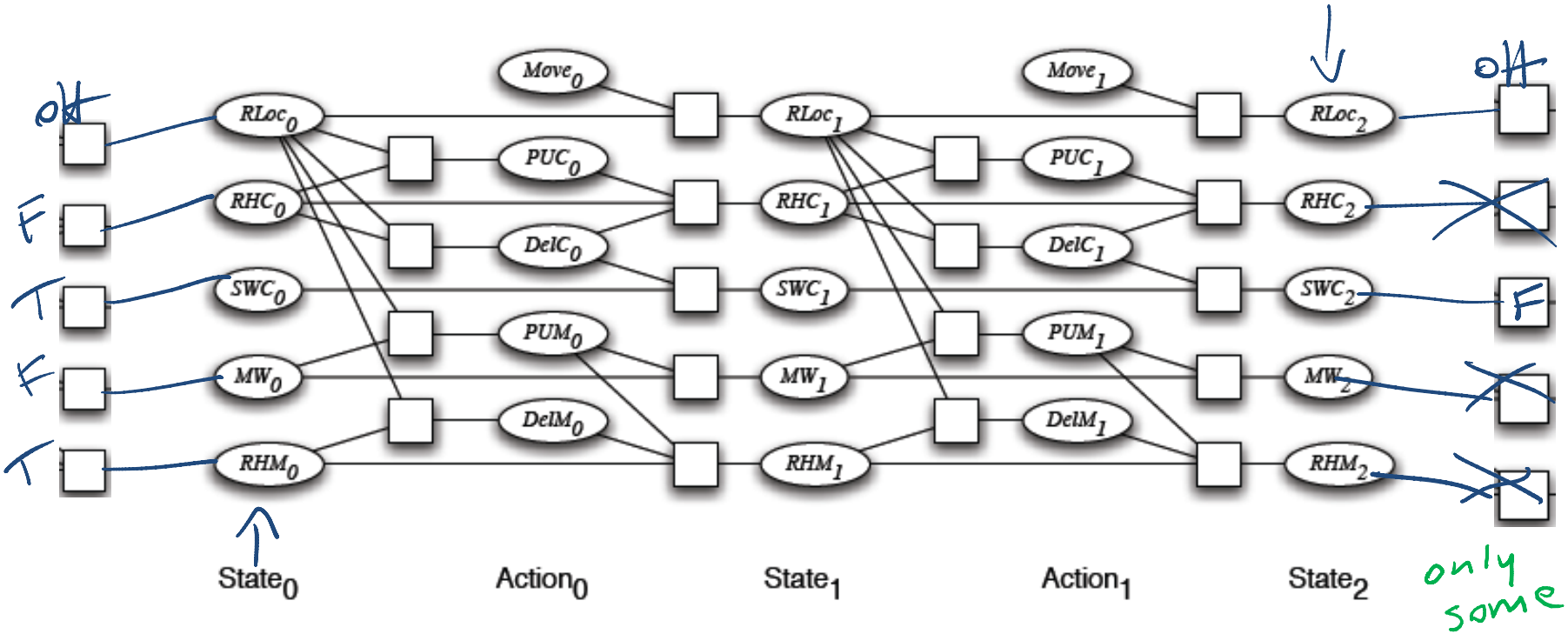
Variables for actions ….

binary

action (non) occurring at that step

CSP Planning: Initial and Goal Constraints

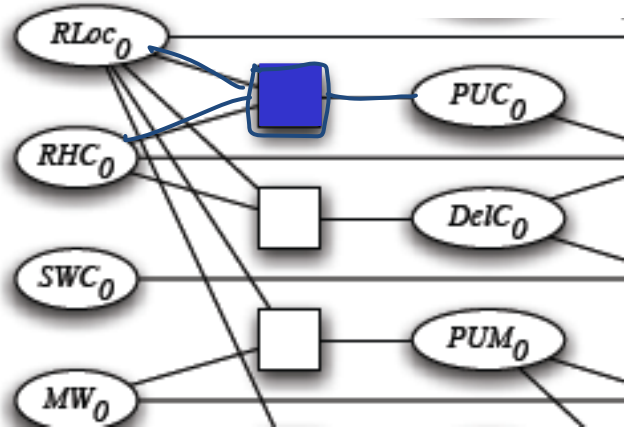
- initial state constraints constrain the state variables at time 0
- goal constraints constrain the state variables at time *k*



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

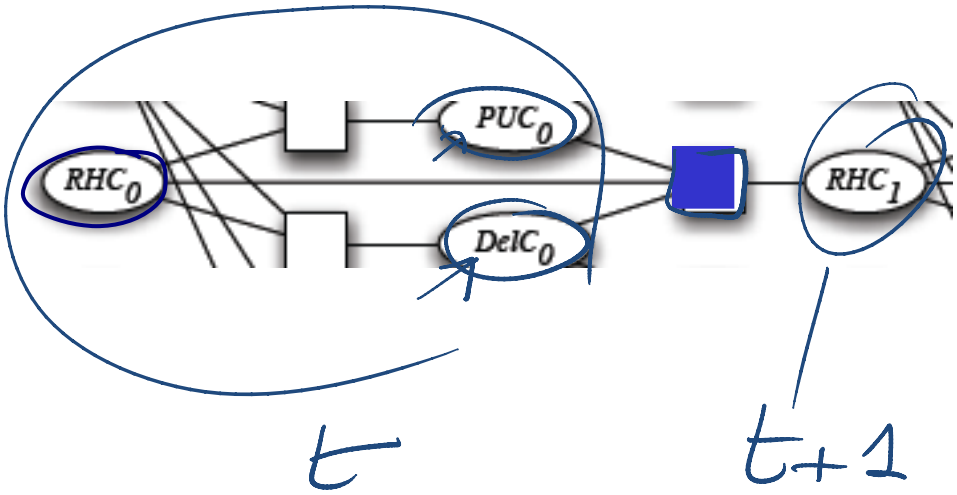
Rob has coffee

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

PUC_0 (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	T
T	T	F	F
T	F	T	T
…	…	…	…
…	…	…	…

CSP Planning: Constraints Contd.

Other constraints we may want are action constraints:

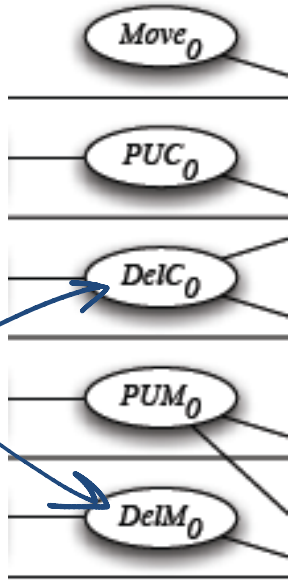
- specify which actions cannot occur simultaneously
- these are sometimes called mutual exclusion (mutex) constraints

E.g., in the Robot domain

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

But we could change that…

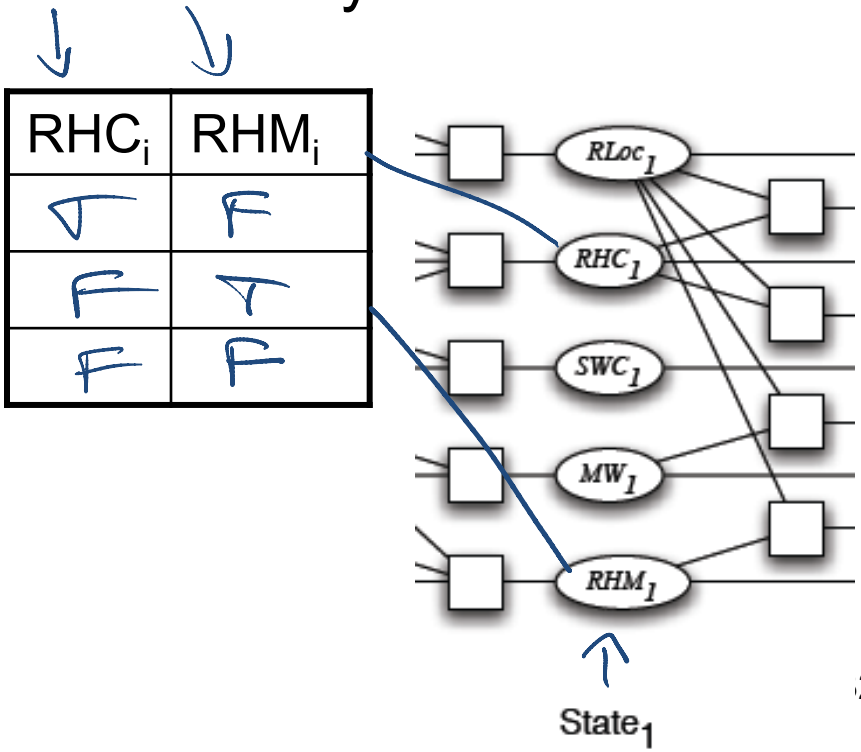
$DelM_i$	$DelC_i$
T	F
F	T
F	F



CSP Planning: Constraints Contd.

Other constraints we may want are state constraints

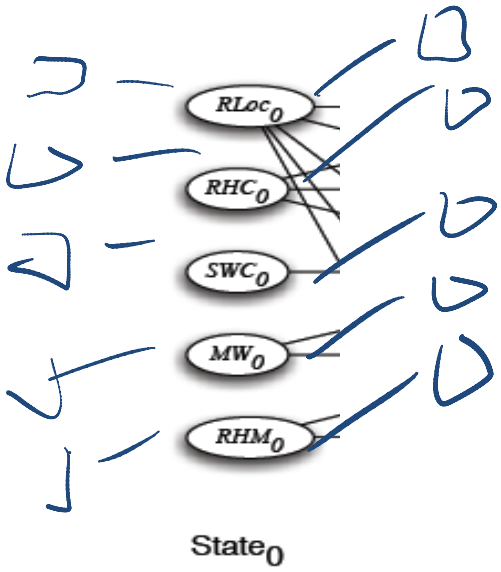
- hold between variables at the same time step
- they can capture physical constraints of the system (robot cannot hold coffee and mail)
- they can encode maintenance goals



CSP Planning: Solving the problem

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

Run arc consistency and search or stochastic local search!



k = 0

Is $State_0$ a goal?

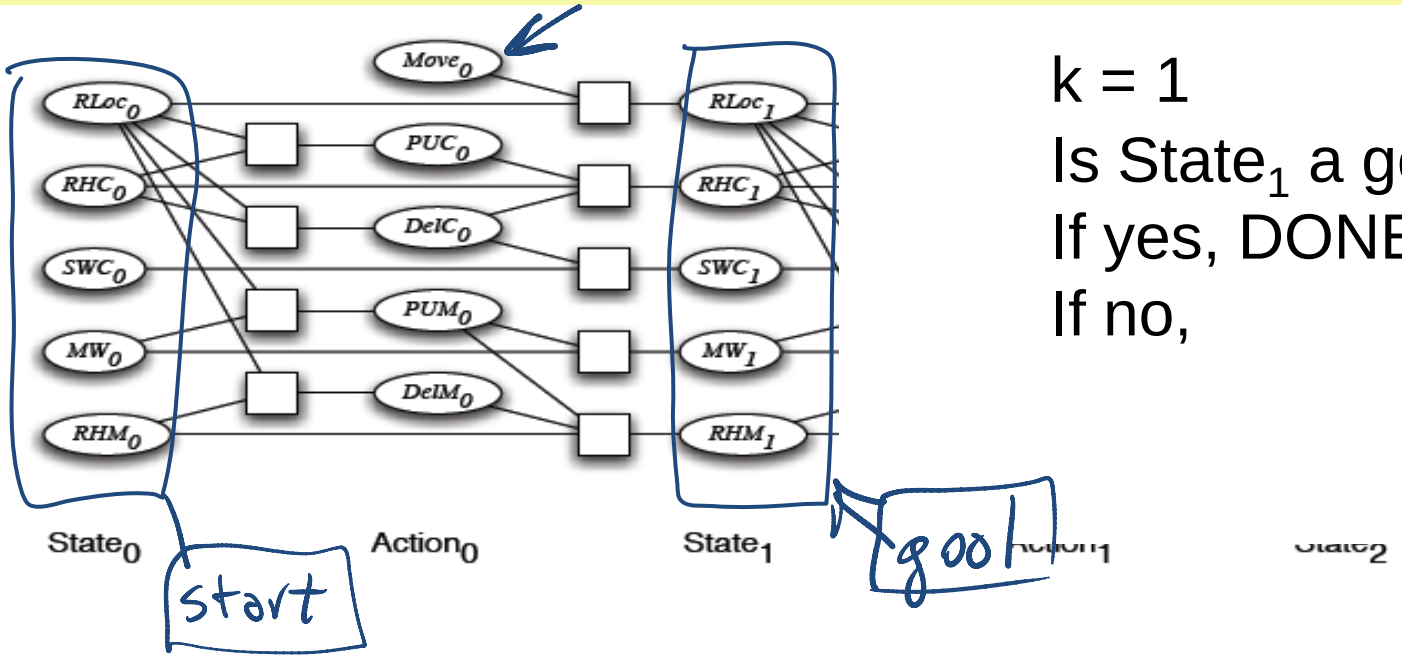
If yes, DONE!

If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon k =1

Run arc consistency and search or stochastic local search!



k = 1

Is $State_1$ a goal

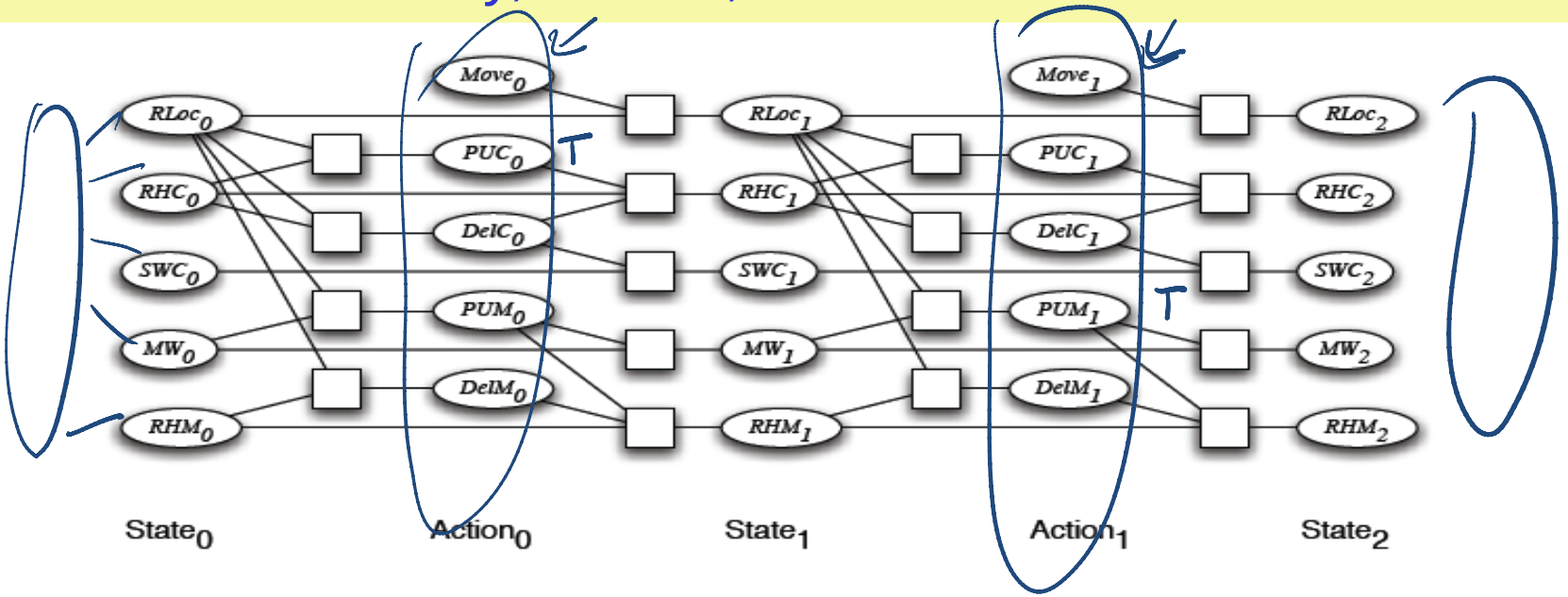
If yes, DONE!

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

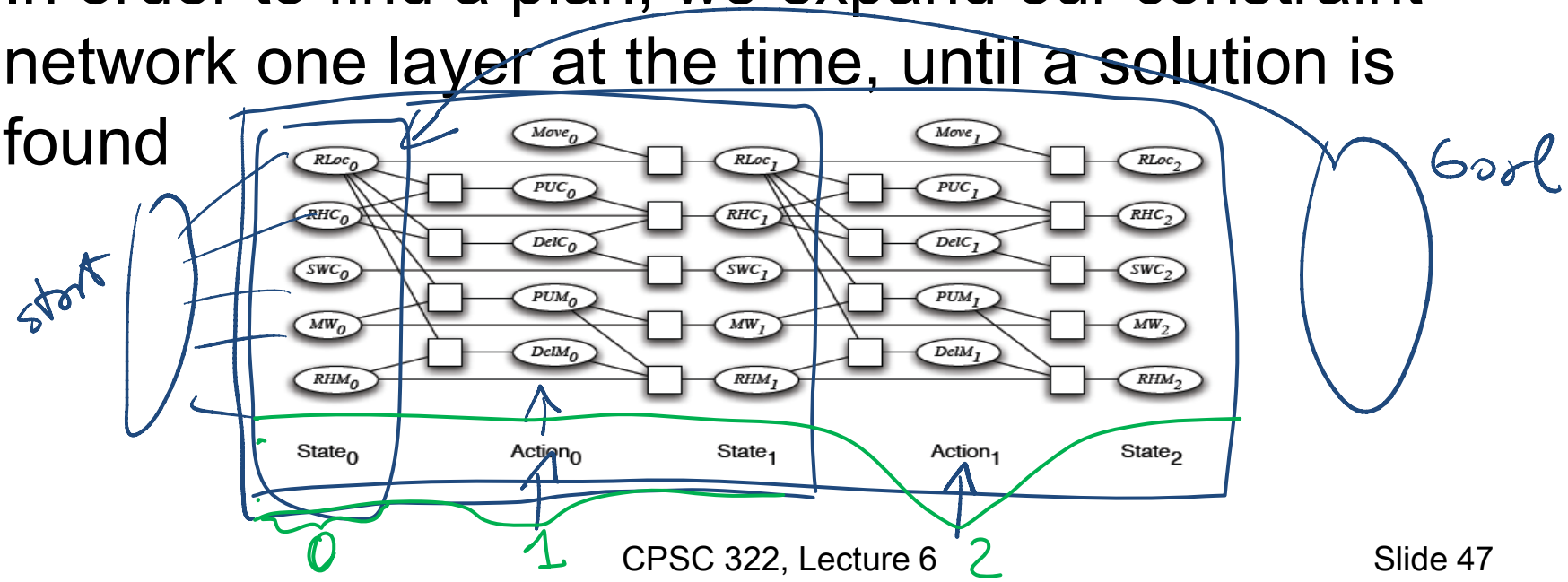
CSP Planning: Solving the problem

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



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

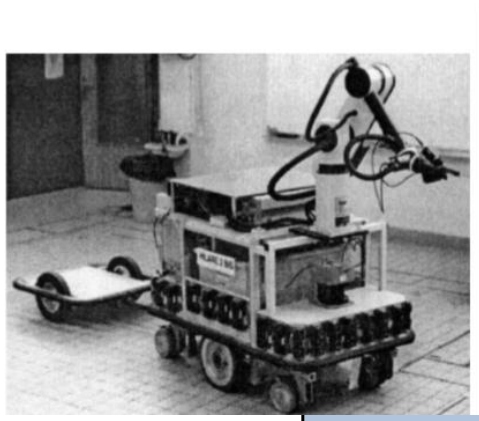
Practice exercise using STRIPS to CSP is available on AIspace

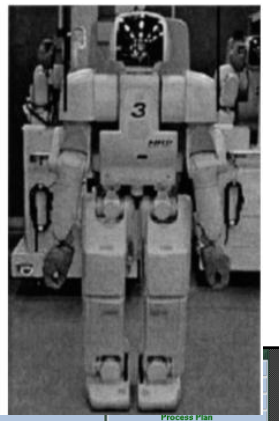
Learning Goals for planning – part-2

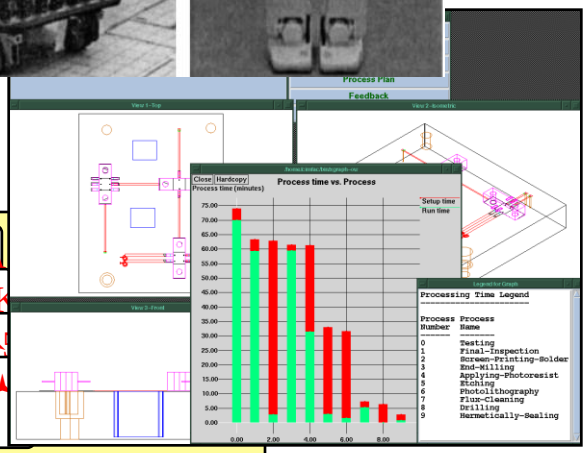
- Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
- Solve a planning problem with CPS by expanding the horizon (new one)

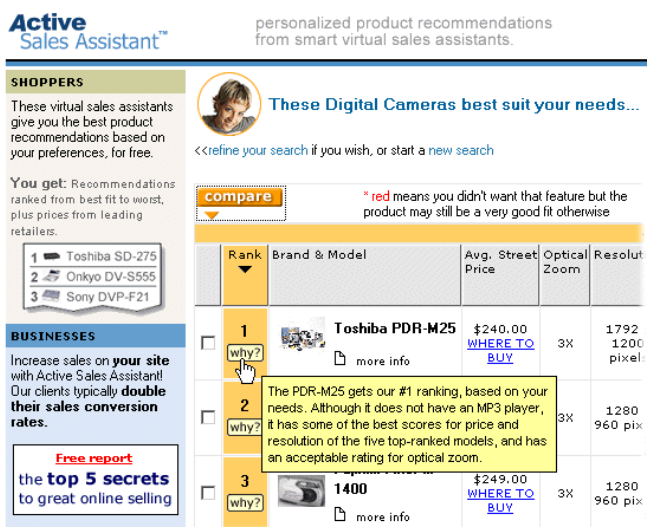
Now, do you know how to implement a planner for….

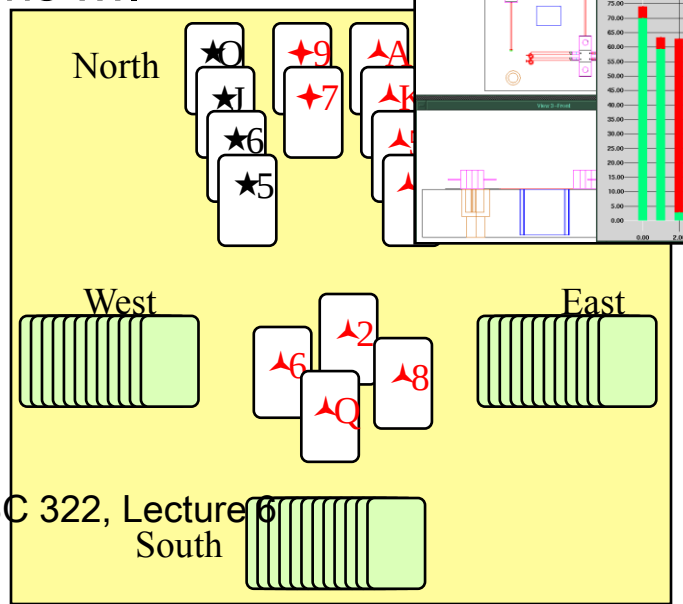
- Emergency Evacuation?
- Robotics?
- Space Exploration?
- Manufacturing Analysis?
- Games (e.g., Bridge)?
- Generating Natural language
  - Product Recommendations ….

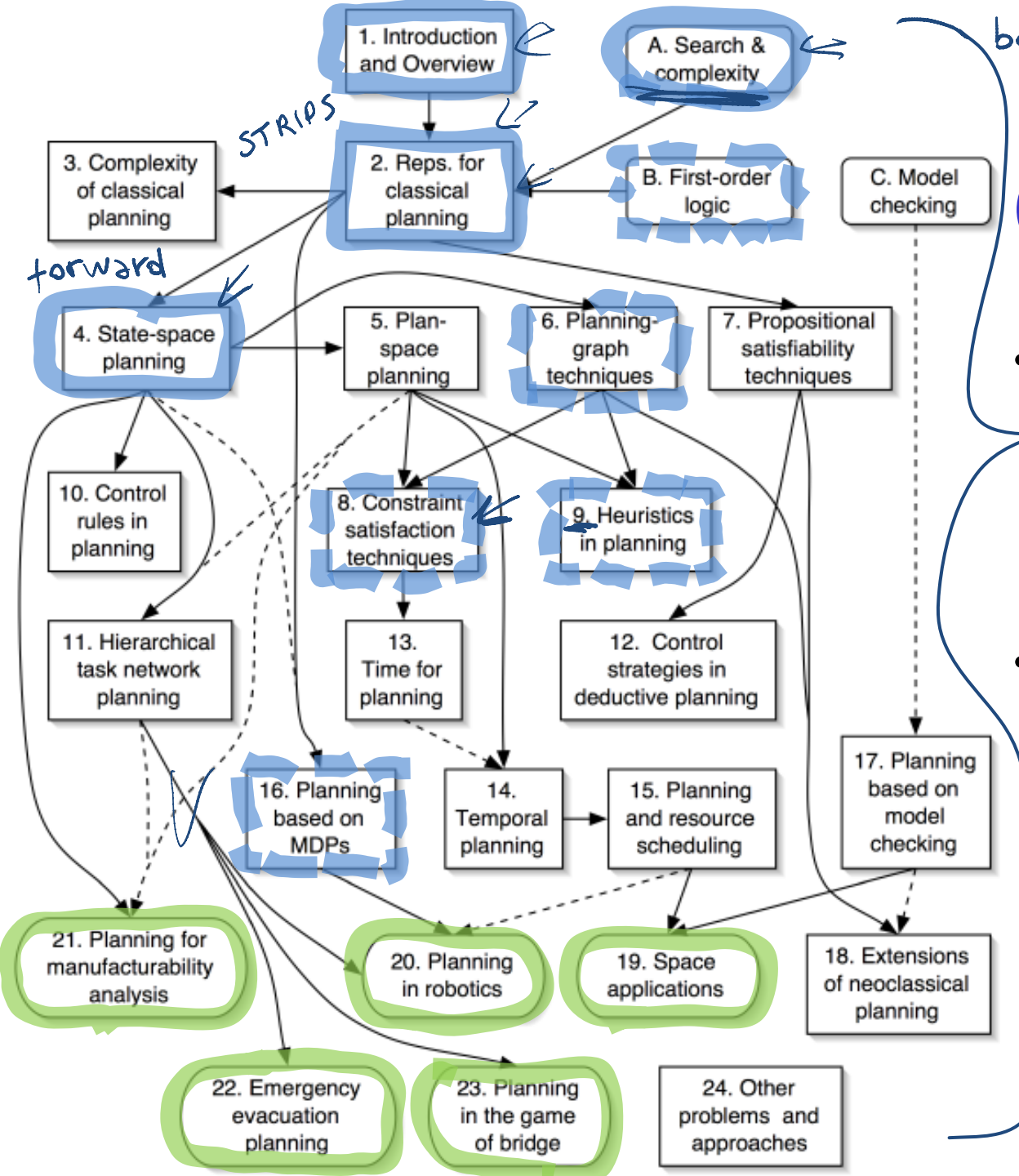












book chapters

No ☹, but you (will) know the key ideas ☺!

- Ghallab, Nau, and Traverso
  Automated Planning: Theory and Practice
  Morgan Kaufmann, May 2004
  ISBN 1-55860-856-7
- Web site:
  - ✓ http://www.laas.fr/planning

you know

you know a little

Applications

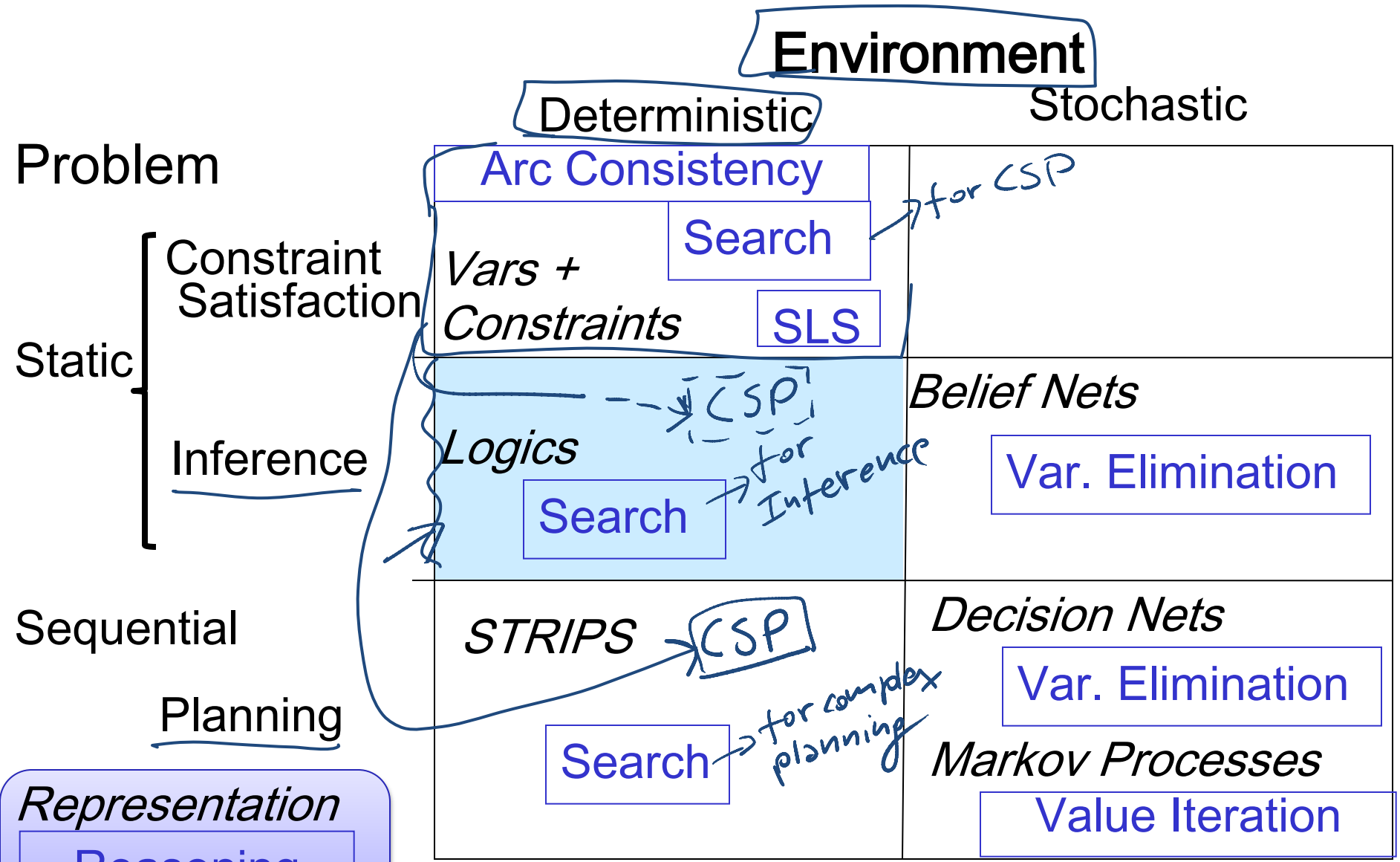
Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

- **Logic Intro** (Chp 5.1- 5.1.1)
  - Propositional Definite Clause Logic: Syntax

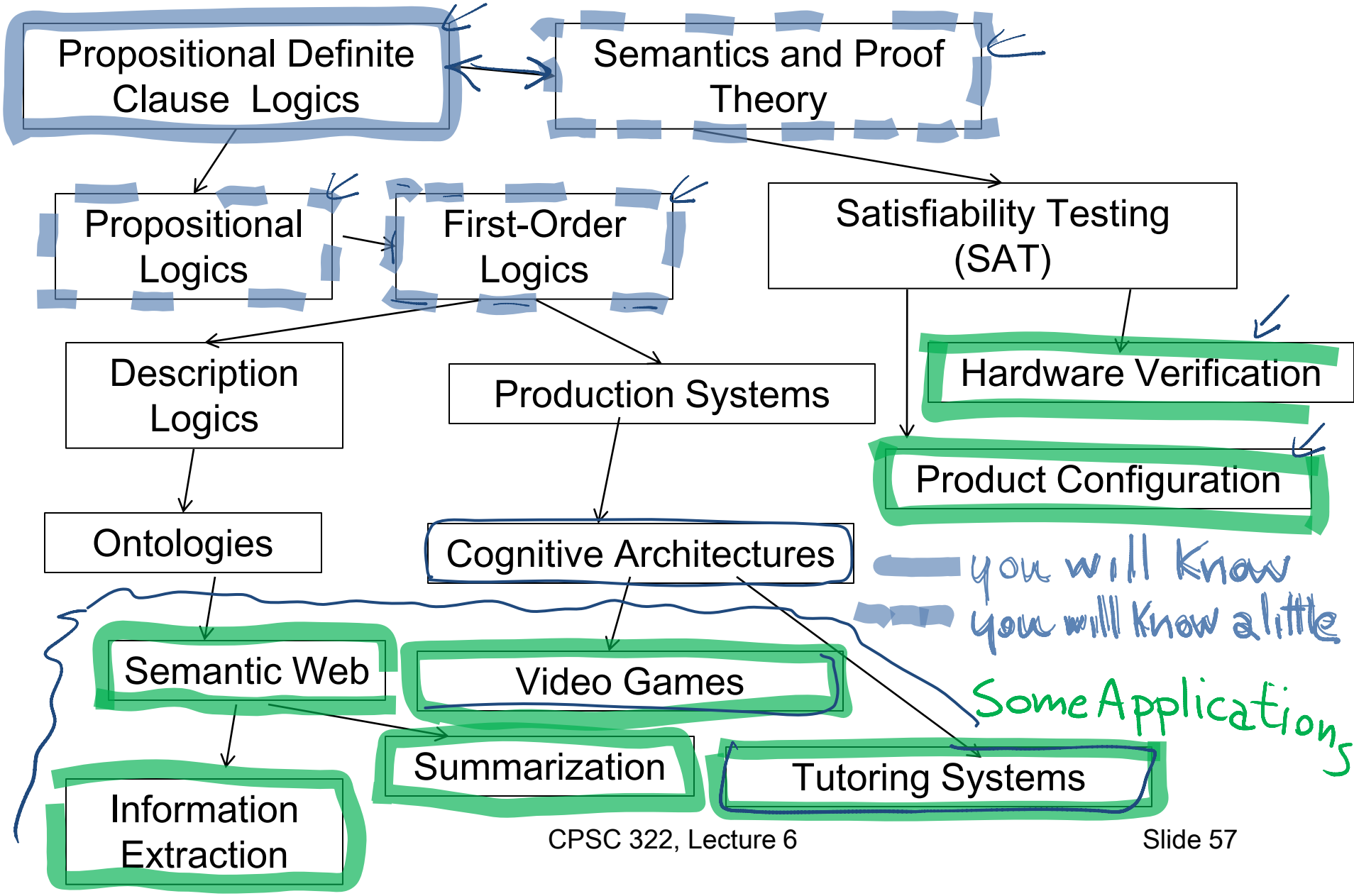
Modules we'll cover in this course: R&Rsys



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
- ……

Logics in AI: Similar slide to the one for planning



What you already know about logic...

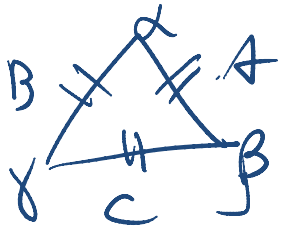
From programming: Some logical operators

```
If ((amount > 0) && (amount < 1000)) || !(age < 30)
...
```

AND ∧ OR ∨ NOT ~, ¬

You know what they mean in a "procedural" way

Logic is the language of Mathematics. To define formal structures (e.g., sets, graphs) and to proof statements about those



$\forall x \; triangle(x) \; A=B=C \iff \alpha=\beta=\gamma$

We are going to look at Logic as a **Representation and Reasoning System** that can be used to **formalize a domain (e.g., an electrical system, an organization)** and to **reason about it**

Logic: A general framework for representation & reasoning

- Let's now think about **how to represent an environment** about which we have only partial (but certain) information
- What do we need to represent?

objects

events

actions

space

time

Why Logics?

- **"Natural"** to express **knowledge** about the world

(more natural than a "flat" set of variables & constraints)

"Every 322 student will pass the midterm"

$Midterm(m_1)$

$Course(c_1)$

$Name\text{-}of(c_1, 322)$

$Course\text{-}of(m_1, c_1)$

$\forall z \; Student(z) \wedge Registred(z, c_1) \wedge Follows\_advice(z, Slide\,23) \Rightarrow pass(m_1, z)$

- It is easy to incrementally add knowledge
- It is easy to check and debug knowledge
- Provide language for asking complex queries
- Well understood formal properties

Propositional Logic

We will study the simplest form of Logic: Propositional

- The primitive elements are **propositions**: Boolean variables that can be {*true, false*}

p_1 p_2

- The goal is to illustrate the basic ideas
- This is a starting point for more complex logics (e.g., first-order logic)
- Boolean nature can be exploited for efficiency.

Propositional logic: Complete Language

The **proposition** symbols p_1, p_2 … etc are sentences

- If S is a sentence, **¬S** is a sentence (negation)
- If S_1 and S_2 are sentences, $\mathbf{S_1 \wedge S_2}$ is a sentence (conjunction)
- If S_1 and S_2 are sentences, $\mathbf{S_1 \vee S_2}$ is a sentence (disjunction)
- If S_1 and S_2 are sentences, $\mathbf{S_1 \Rightarrow S_2}$ is a sentence (implication)
- If S_1 and S_2 are sentences, $\mathbf{S_1 \Leftrightarrow S_2}$ is a sentence (biconditional)

Sample Formula

$$((p_1 \vee p_2) \wedge p_3) \Leftrightarrow ((p_2 \Rightarrow \neg p_4) \vee p_5)$$

Propositional Logics in practice

- Agent is told (perceives) some facts about the world
  some propositions are true
- Agent is told (already knows / learns) how the world works
  logical formulas
- Agent can answer yes/no questions about whether other facts must be true

Using Logics to make inferences…

1) Begin with a **task domain**.

2) Distinguish those **things** you want to talk about (the ontology).

SEE NEXT SLIDE

3) Choose symbols in the computer to **denote propositions**

$live\_w_6$ sw_1_on

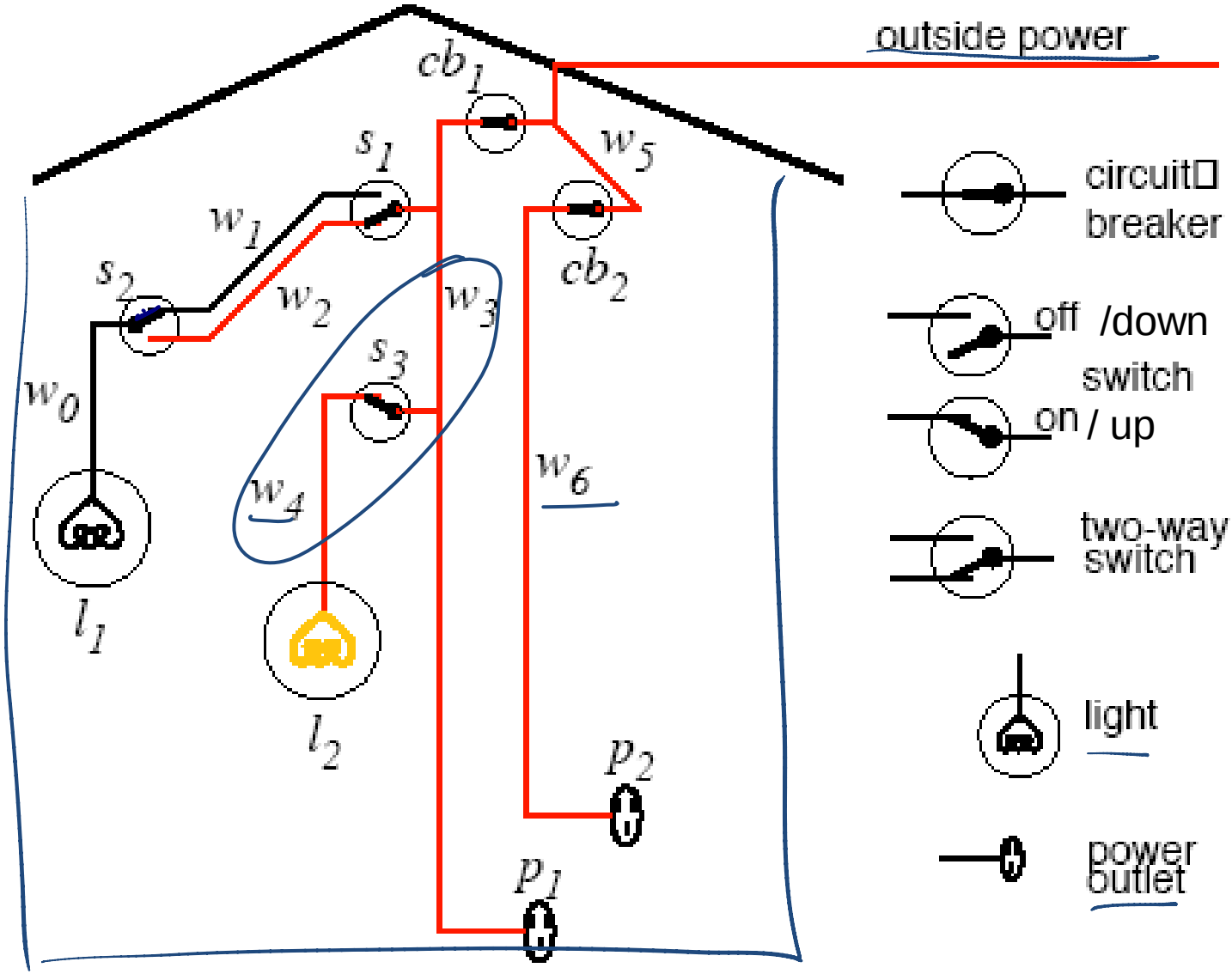
4) Tell the system **knowledge** about the domain.

$live\_w_3 \wedge sw_3\_on \rightarrow live\_w_4$

5) **Ask the system** whether new statements about the domain are true or false.

l_2_on ?

Electrical Environment



Lecture Overview

Planning

- Example
- STRIPS: a Feature-Based Representation
- Forward Planning
- Heuristics
- CSP Planning

- **Logic Intro**
  - Propositional Definite Clause Logic: Syntax (Chp 5.2)

Propositional Definite Clauses

- **Propositional Definite Clauses:** our first logical representation and reasoning system.

(very simple!)

- Only two kinds of statements:
  - that **a proposition is true**
  - that **a proposition is true if one or more other propositions are true**

p_1 p_2

$p_1 \leftarrow p_3 \wedge p_4$

- Why still useful?
  - Adequate in many domains (with some adjustments)
  - Reasoning steps easy to follow by humans
  - Inference linear in size of your set of statements
  - Similar formalisms used in cognitive architectures

Propositional Definite Clauses: Syntax

Definition (atom)

An **atom** is a symbol starting with a lower case letter

Definition (body)

A **body** is an atom or is of the form $b_1 \wedge b_2$ where b_1 and b_2 are bodies.

Definition (definite clause)

A **definite clause** is an atom or is a rule of the form $h \leftarrow b$ where h is an atom and b is a body. (Read this as ``h if b.")

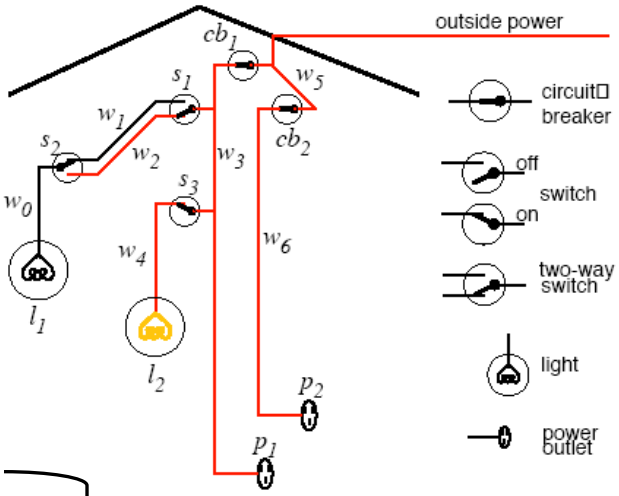
Definition (KB)

A **knowledge base** is a set of definite clauses

definite clauses, KB

atoms

light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.



rules

live_l1 ←live_w0.
live_w0 ←live_w1 ∧up_s2.
live_w0 ←live_w2 ∧down_s2.
live_w1 ←live_w3 ∧up_s1.
live_w2 ←live_w3 ∧down_s1.
live_l2 ←live_w4.
live_w4 ←live_w3 ∧up_s3.
live_p1 ←live_w3.
live_w3 ←live_w5 ∧ok_cb1.
live_p2 ←live_w6.
live_w6 ←live_w5 ∧ok_cb2.
live_w5 ←live_outside.
lit_l1 ←light_l1 ∧live_l1 ∧ok_l1.
lit_l2 ←light_l2 ∧live_l2 ∧ok_l2.

PDC Syntax: more examples

Definition (definite clause)

A **definite clause** is

- an atom or
- a rule of the form $h \leftarrow b$ where h is an atom ('head') and b is a body. (Read this as 'h if b.')

Legal PDC clause

Not a legal PDC clause

a) *ai_is_fun*

b) *ai_is_fun v ai_is_boring*

c) *ai_is_fun ← learn_useful_techniques*

d) *ai_is_fun ← learn_useful_techniques ∧ notTooMuch_work*

e) *ai_is_fun ← learn_useful_techniques ∧¬ TooMuch_work*

f) *ai_is_fun ← f(time_spent, material_learned)*

g) *srtsyj ← errt ∧ gffdgdgd*

PDC Syntax: more examples

Legal PDC clause

Not a legal PDC clause

a) *ai_is_fun*

b) *ai_is_fun ∨ ai_is_boring*

c) *ai_is_fun ← learn_useful_techniques*

d) *ai_is_fun ← learn_useful_techniques ∧ notTooMuch_work*

e) *ai_is_fun ← learn_useful_techniques ∧ ¬ TooMuch_work*

f) *ai_is_fun ← f(time_spent, material_learned)*

g) *srtsyj ← errt ∧ gffdgdgd*

Do any of these statements mean anything?
Syntax doesn't answer this question!

Learning Goals for today's: Logic

You can:

- Verify whether a logical statement belongs to the language of full propositional logics.
- Verify whether a logical statement belongs to the language of propositional definite clauses.

Study for midterm (Mon May 28 – 3PM - rm 310)

Midterm: ~6 short questions (*10pts each*) + 2 problems (*20pts each)*

- Study: textbook and **inked** slides
- Work on **all** practice exercises and **revise assignments**!
- While you revise the **learning goals**, work on **review questions** (will post them tomorrow)- I may even reuse some verbatim ☺
- Will post a **couple of problems** from previous offering (maybe slightly more difficult /inappropriate for you because they were not informed by the learning goals) … but I'll give you the solutions ☺

Next class

- Definite clauses Semantics and Proofs (textbook 5.1, 5.2)