# Planning: Wrap up CSP Planning
# Logic: Intro

Alan Mackworth

UBC CS 322 - Planning 3

February 25, 2013

Textbook §8.4, §5.1

# GAC vs. AC

- GAC = Generalized Arc Consistency (for k-ary constraints)
- AC = Arc Consistency for Binary Arcs
- In class we focused on AC, text covers GAC

  (You are responsible for text sections covered.)

# Arc consistency algorithm (for binary constraints)

**Procedure** GAC(V,dom,C)

**Inputs**

V: a set of variables
dom: a function such that dom(X) is the domain of variable X
C: set of constraints to be satisfied

**Output**

arc-consistent domains for each variable

**Local**

$D_X$ is a set of values for each variable X
TDA is a set of arcs

**for each** variable X **do**
    $D_X \leftarrow \text{dom}(X)$
TDA $\leftarrow \{ \langle X,c \rangle \mid X \in V, c \in C$ and $X \in \text{scope}(c)\}$

**while** (TDA $\neq \{\}$)
    **select** $\langle X,c \rangle \in$ TDA
    TDA $\leftarrow$ TDA $\setminus \{ \langle X,c \rangle \}$
    $ND_X \leftarrow \{x \mid x \in D_X$ and $\exists\, y \in D_Y$ s.t. $(x, y)$ satisfies $c\}$
    **if** ($ND_X \neq D_X$) **then**
        TDA $\leftarrow$ TDA $\cup \{ \langle Z,c' \rangle \mid X \in \text{scope}(c'), c' \neq c, Z \in \text{scope}(c') \setminus \{X\} \}$
        $D_X \leftarrow ND_X$

**return** $\{D_X \mid X$ is a variable$\}$

# GAC for k-ary constraints (P&M, Section 4.5)

```
Procedure GAC(V,dom,C)
        Inputs
                V: a set of variables
                dom: a function such that dom(X) is the domain of variable X
                C: set of constraints to be satisfied
        Output
                arc-consistent domains for each variable
        Local
                D_X is a set of values for each variable X
                 TDA is a set of arcs
         for each variable X do
                 D_X ←dom(X)
         TDA ←{⟨X,c⟩| c∈C  and X∈scope(c)}
         while (TDA ≠{})
                 select ⟨X,c⟩ ∈TDA;
                 TDA ←TDA  \ {⟨X,c⟩};
                 ND_X ←{x| x ∈D_X and some {X=x,Y_1=y_1,...,Y_k=y_k}∈c where y_i ∈D_Yi for all i}
                 if (ND_X ≠D_X) then
                         TDA ←TDA ∪{⟨Z,c'⟩|X ∈scope(c'), c' is not c, Z∈scope(c') \ {X} }
                         D_X ←ND_X
         return {D_X| X  is a variable}
```

Figure 4.3: Generalized arc consistency algorithm

# Lecture Overview

- Recap: STRIPS, forward planning & heuristics
- Recap: CSP planning
- More CSP planning
  - Details on CSP representation
  - Solving the CSP planning problem
- Time permitting: Intro to Logic

# Course Overview

**Course Module:** *Representation* — Reasoning Technique

**Environment**

**Problem Type**

| Problem Type | | Deterministic | Stochastic | |
|---|---|---|---|---|
| Static | Constraint Satisfaction | ***Variables + Constraints*** — Arc Consistency; Search | | |
| Static | Logic | ***Logics*** — Search | ***Bayesian Networks*** — Variable Elimination | Uncertainty |
| Sequential | Planning | ***STRIPS*** — Search; As CSP (using arc consistency) | ***Decision Networks*** — Variable Elimination; ***Markov Processes*** — Value Iteration | Decision Theory |

We're here: deterministic & sequential

# STRIPS

Definition:

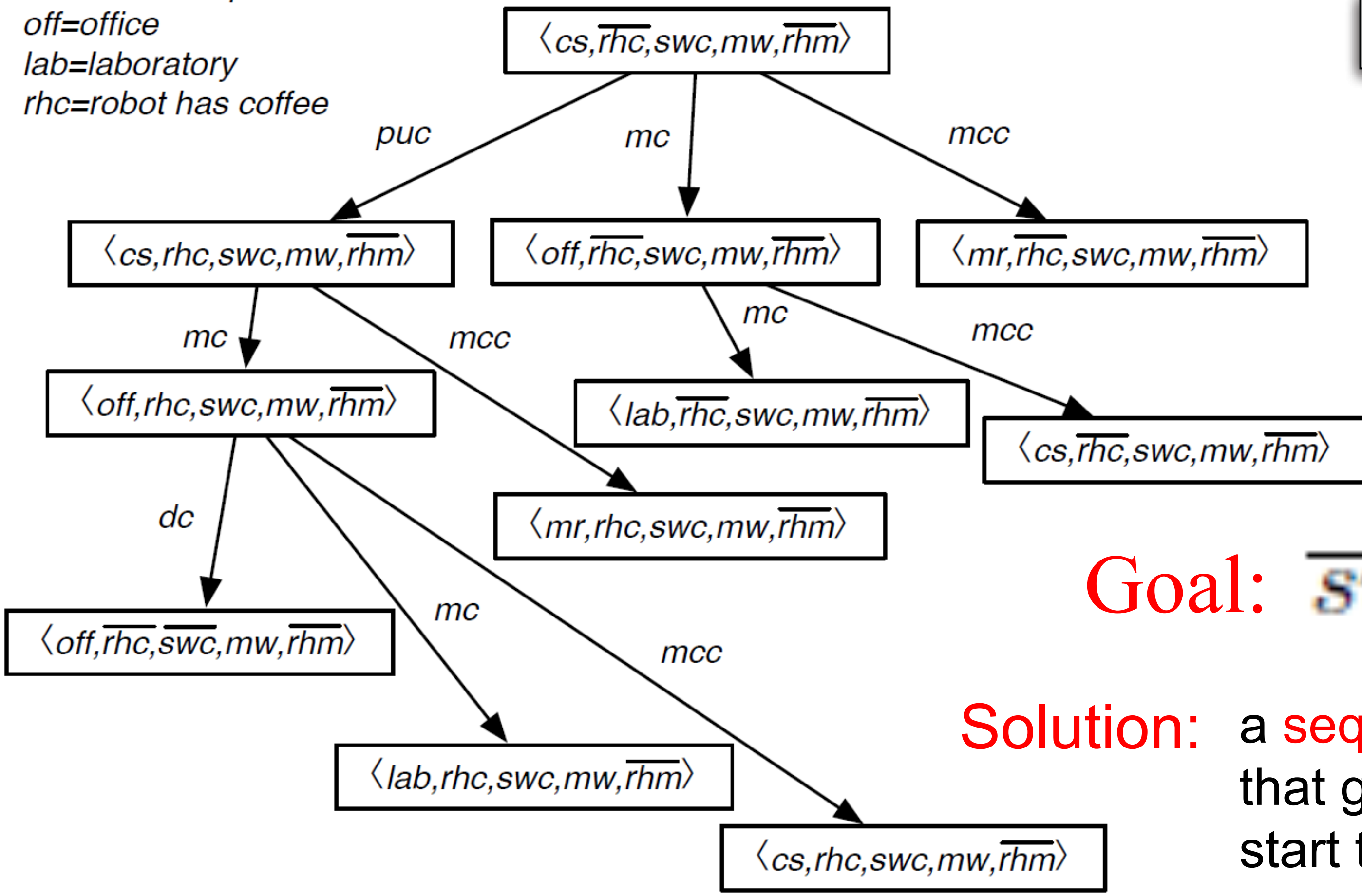
A STRIPS problem instance consists of:

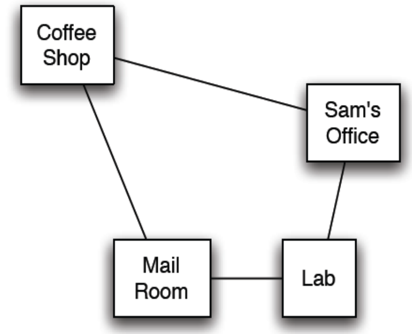
- a set of variables (features) $\mathcal{V}$
- a domain dom(V) for each variable V $\in \mathcal{V}$
  - Let $\mathcal{X}$ be the space of partial assignments of a set of variables to values from their domains
- a set of actions $\mathcal{A}$
  - Each action a$\in\mathcal{A}$ has
    - A set of preconditions P(a)$\in\mathcal{X}$
    - A set of effects E(a)$\in\mathcal{X}$
- a start condition s $\in \mathcal{X}$
- a goal condition g $\in \mathcal{X}$

- Example for an action in robot example: pick up coffee
  - preconditions Loc = cs and RHC = $\overline{rhc}$
  - effects RHC = rhc

# Forward planning: search in 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, dc)

# Planning as Standard Search

- 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: relaxed problem! E.g. "ignore delete lists"
- Inference
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function

# Example for domain-independent heuristics

- Let's stay in the robot domain
  - But say our robot has to bring coffee to Bob, Sue, and Steve:
  - G = {bob_has_coffee, sue_has_coffee, steve_has_coffee}
  - They all sit in different offices
- Admissible heuristic 1: ignore preconditions:
  - Basically counts how many subgoals are not achieved yet
  - Can simply apply "DeliverCoffee(person)" action for each person
- Admissible heuristic 2: ignore "delete lists"
  - Rewrite effects as add and delete lists, e.g.:
    - Add list for "pick-up coffee": rhc
    - Delete list for "deliver coffee": rhc
  - Here: "Ignore delete lists" ⇔ once you have coffee you keep it
    - Problem gets easier: only need to pick up coffee once, navigate to the right locations, and deliver
    - Admissible, but typically more realistic than ignoring preconditions

# Lecture Overview

- Recap: STRIPS, forward planning & heuristics
- Recap: CSP planning
- More CSP planning
  - Details on CSP representation
  - Solving the CSP planning problem
- Time permitting: Intro to Logic

# Planning as a CSP

- Idea: reformulate a STRIPS model as a set of variables and constraints
- What are variables in the CSP? (more than one answer is correct)

The STRIPS variables

The values of the STRIPS variables

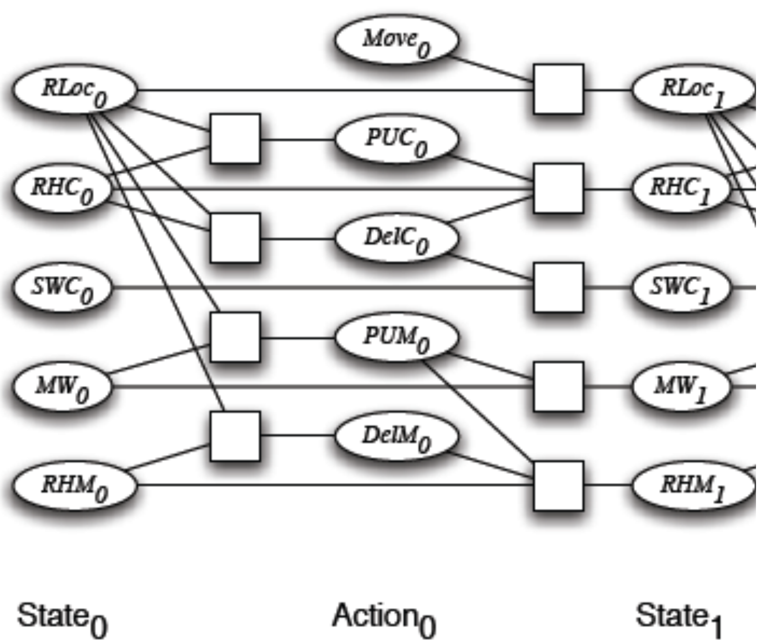
The STRIPS actions

The STRIPS preconditions

- We have CSP variables for both
  - STRIPS variables and
  - STRIPS actions

# Planning as a CSP: General Idea

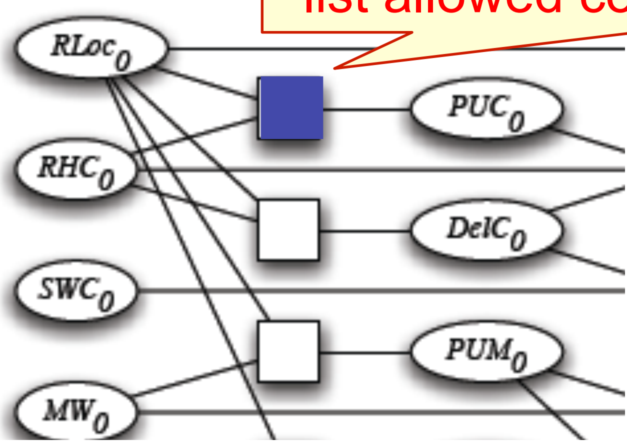
- Both features and actions are CSP variables
  - One CSP variable for each STRIPS feature for each time step
  - One (Boolean) CSP variable for each time step for each action
- Main Constraints:
  – Between actions at time t and previous state variables (time t)
    - When does an action apply? (precondition constraints)
  – Between actions at time t and following state variables (time t+1)
    - How does an action change the variables? (effect constraints)



# CSP Planning: Precondition Constraints

- precondition constraints
  - 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 Loc=cs (coffee shop) and RHC = false (don't have coffee already)

Truth table for this constraint:
list allowed combinations of values



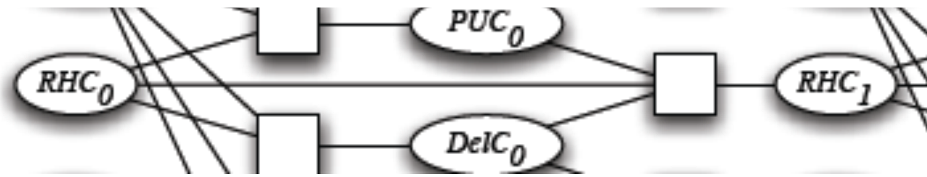

| $RLoc_0$ | $RHC_0$ | $PUC_0$ |
|---|---|---|
| cs | T | F |
| cs | F | T |
| cs | F | F |
| mr | * | F |
| lab | * | F |
| off | * | F |

Need to allow for the option of *not* taking an action even when it is valid

# CSP Planning: Effect Constraints

- Effect constraints
  - Between action variables at time t and state variables at time t+1
  - Specify the effects of an action
  - Also depends on state variables at time t (frame rule!)
    - E.g. let's consider RHC at time t and t+1
      Let's fill in a few rows in this table

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



# Planning as a CSP

- What gives rise to constraints in the CSP? (more than one answer is correct)

The STRIPS preconditions

The STRIPS effects

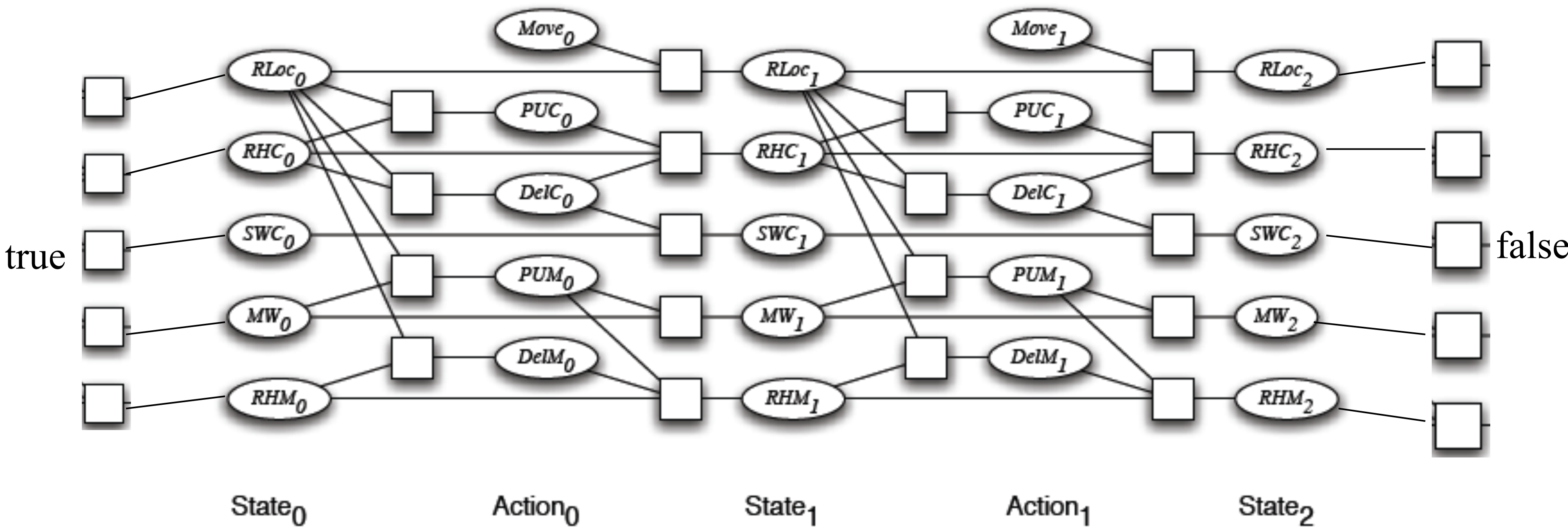
The STRIPS start state

The STRIPS goal condition

- All of them!
- Plus, constraints between each variable V at time t and t+1:
  - If no action changes V, it stays the same
  - Called a frame constraint

# Initial and Goal Constraints

- initial state constraints: **unary** constraints on the values of the state variables at time 0
- goal constraints: **unary** constraints on the values of the state variables at time *k*
- *E.g. start condition: Sam wants coffee*
  *E.g. goal condition: Sam doesn't want coffee*



# Lecture Overview

- Recap: STRIPS, forward planning & heuristics
- Recap: CSP planning
- More CSP planning
  - Details on CSP representation
  - Solving the CSP planning problem
- Time permitting: Intro to Logic

# Additional constraints in CSP Planning

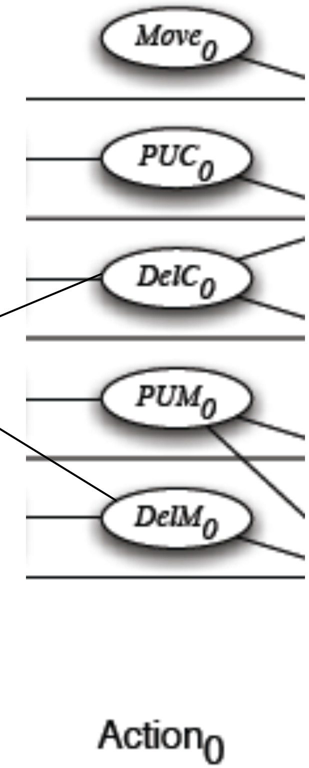
- Other constraints we may want are action constraints:
  - specify which actions cannot occur simultaneously
  - these are often called mutual exclusion (mutex) constraints

E.g. in the Robot domain

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

But we can enforce that they do not happen simultaneously

| $DelM_i$ | $DelC_i$ |
|---|---|
| T | F |
| F | T |
| F | F |

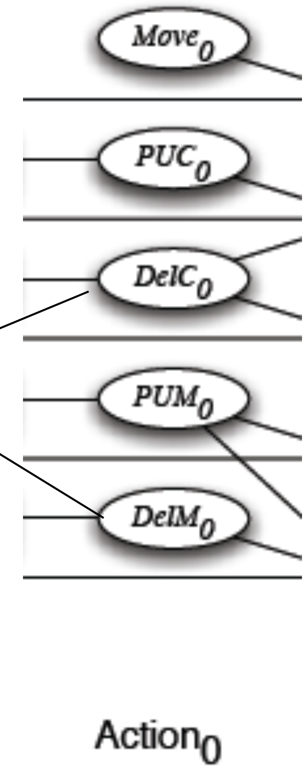


# Handling mutex constraints in Forward Planning

E.g., let's say we don't want *DelM* and *DelC* to occur simultaneously

How would we encode this into STRIPS for forward planning?

| $DelM_i$ | $DelC_i$ |
|---|---|
| T | F |
| F | T |
| F | F |




Via the actions' preconditions (how?)

Via the actions' effects (how?)

No need to enforce this constraint in Forward Planning

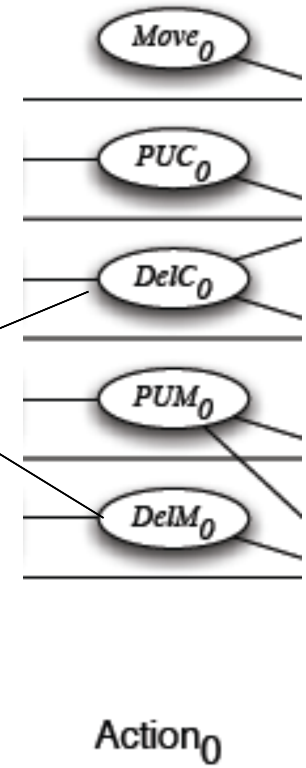
None of the above

# Handling mutex constraints in Forward Planning

E.g., let's say we don't want *DelM* and *DelC* to occur simultaneously

How would we encode this into STRIPS for forward planning?

| $DelM_i$ | $DelC_i$ |
|---|---|
| T | F |
| F | T |
| F | F |




No need to enforce this constraint in Forward Planning

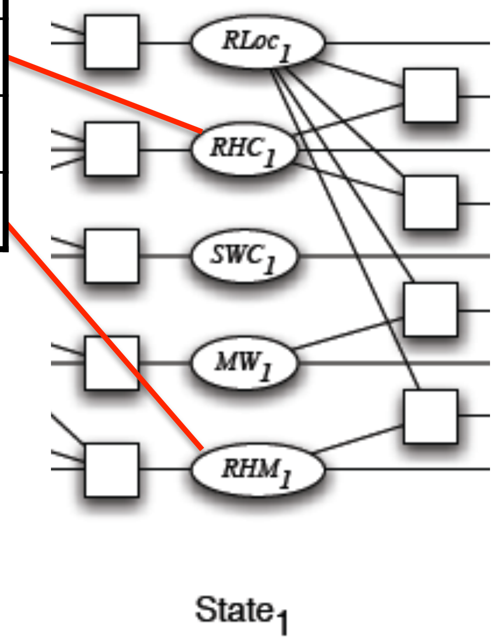
Because forward planning gives us a sequence of actions: only one action is carried out at a time anyways

# Additional constraints in CSP Planning

Other constraints we may want are state constraints

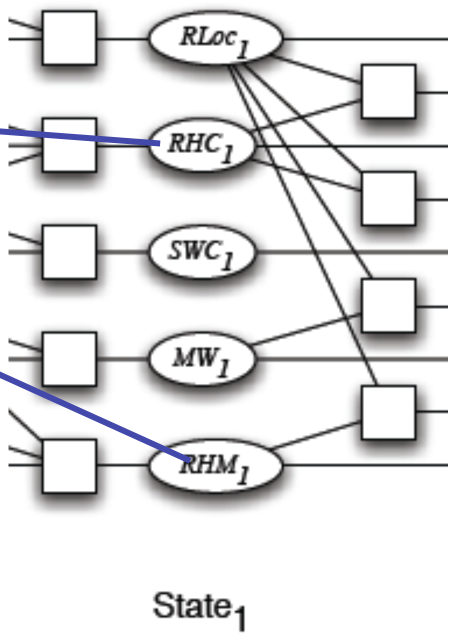
- hold between variables at the same time step
- they can capture physical constraints of the system (e.g., robot cannot hold coffee and mail)

| $RHC_i$ | $RHM_i$ |
|---|---|
| T | F |
| F | T |
| F | F |



# Handling state constraints in Forward Planning

| $RHC_i$ | $RHM_i$ |
|---|---|
| T | F |
| F | T |
| F | F |




How could we handle these constraints in STRIPS for forward planning?

Via the actions' preconditions (how?)

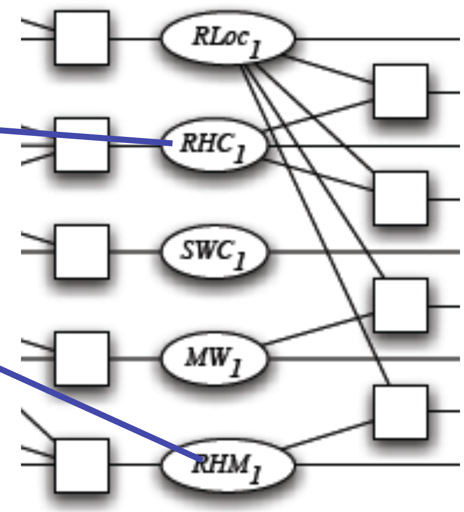
Via the actions' effects (how?)

No need to enforce this constraint in Forward Planning (why?)

None of the above

# Handling state constraints in Forward Planning

| $RHC_i$ | $RHM_i$ |
|---|---|
| T | F |
| F | T |
| F | F |




$State_1$

How could we handle these constraints in STRIPS for forward planning?

We need to use preconditions

- Robot can pick up coffee only if it does not have coffee and it does not have mail
- Robot can pick up mail only if it does not have mail and it does not have coffee

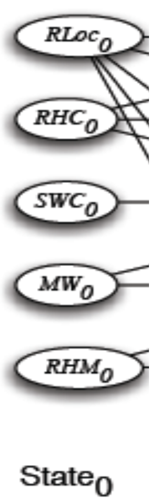
# Lecture Overview

- Recap: STRIPS, forward planning & heuristics
- Recap: CSP planning
- More CSP planning
  - Details on CSP representation
  - Solving the CSP planning problem
- Time permitting: Intro to Logic

# CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0,1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon




K = 0

Is there a solution for this horizon?

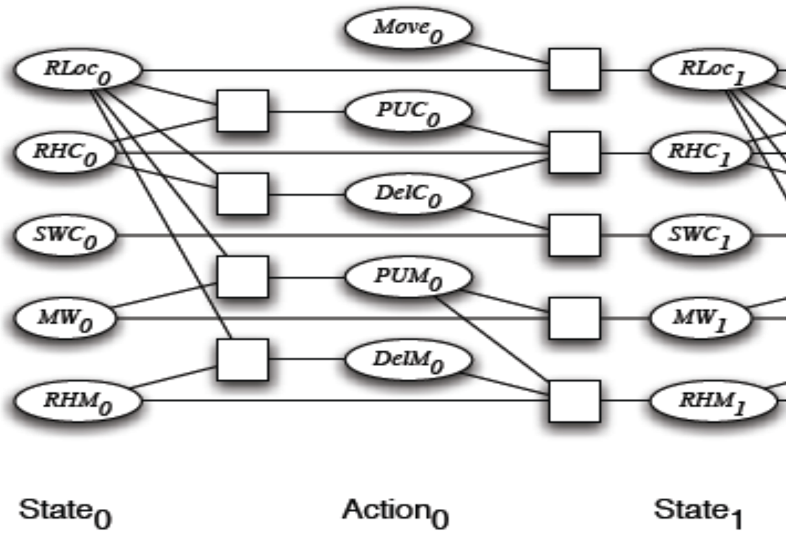
If yes, DONE!

If no, continue …

# CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0,1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon



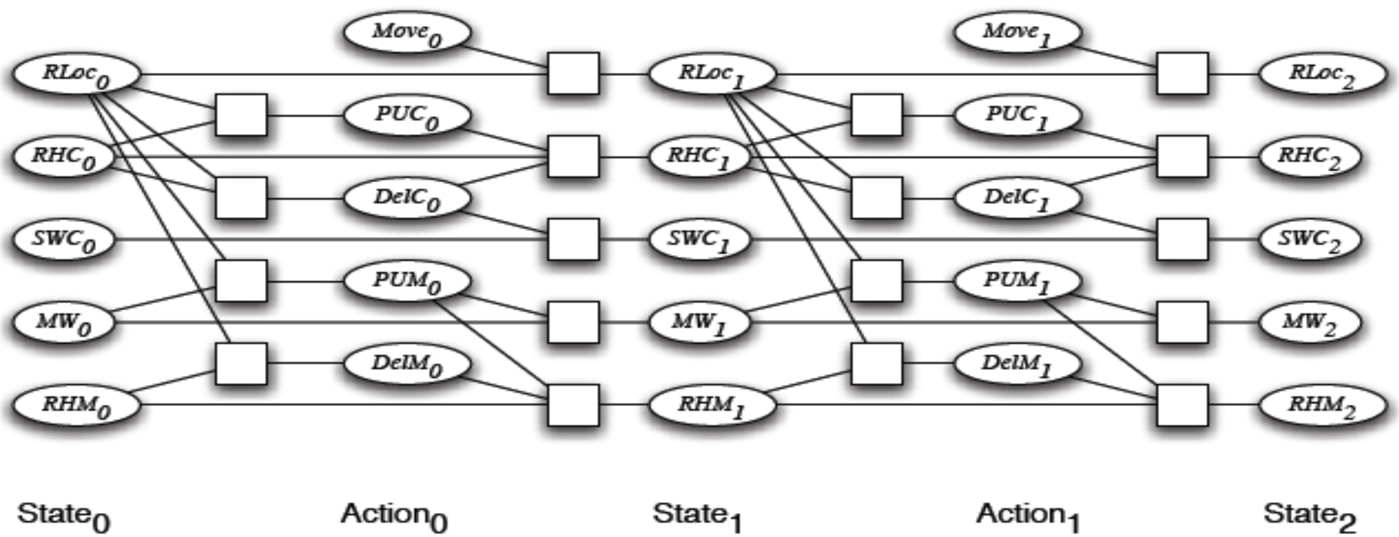

K = 1
Is there a solution for this horizon?

If yes, DONE!
If no, continue …

# CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0,1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon




K = 2: Is there a solution for this horizon?

If yes, DONE!

If no….continue

# Solving Planning as CSP: pseudo code

```
solved = false
for horizon h=0,1,2,…
    map STRIPS into a CSP csp with horizon h
    solve that csp
    if solution exists then
        return solution
    else
        horizon = horizon + 1
end
```

Which method would you use to solve each of these CSPs?

Stochastic Local Search | Arc consistency + domain splitting

Not SLS! SLS cannot determine that no solution exists!

# STRIPS to CSP Conversion 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

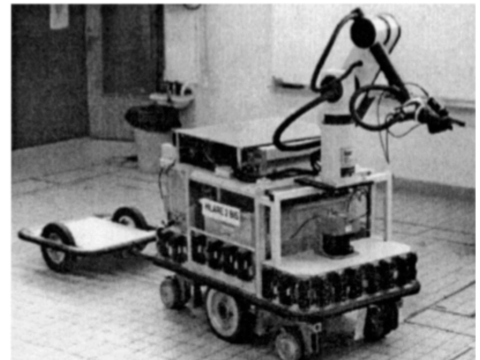
Take it for a spin on sample problems! Assignment #3.

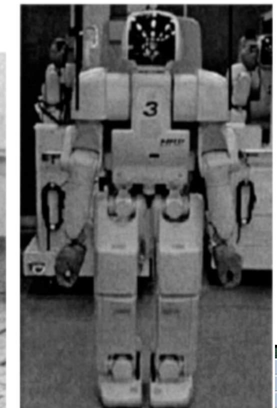
# Learning Goals for Planning

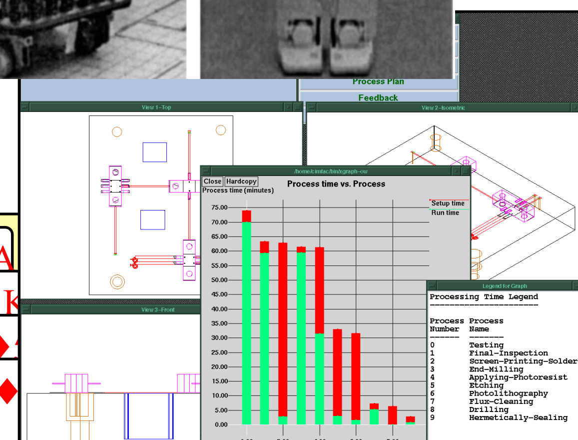
- STRIPS
  - Represent a planning problem with the STRIPS representation
  - Explain the STRIPS assumption
- Forward planning
  - Solve a planning problem by search (forward planning). Specify states, successor function, goal test and solution.
  - Construct and justify a heuristic function for forward planning
- CSP planning
  - Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
  - Solve a planning problem with CSP by expanding the horizon

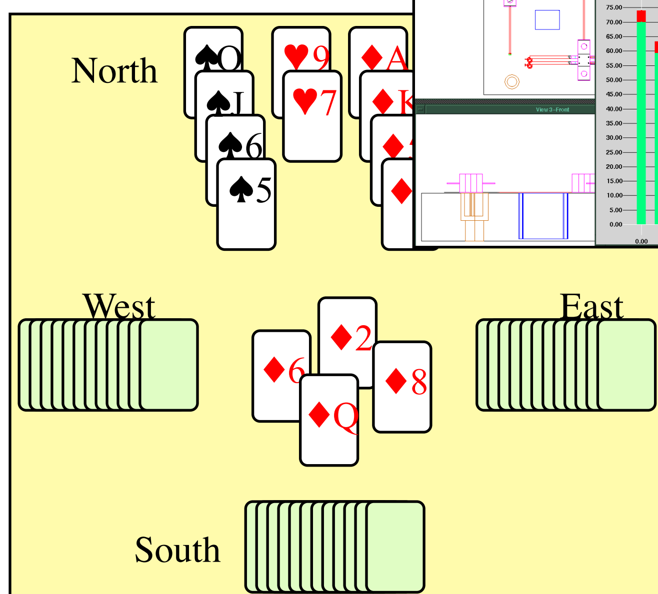
# Some applications of planning

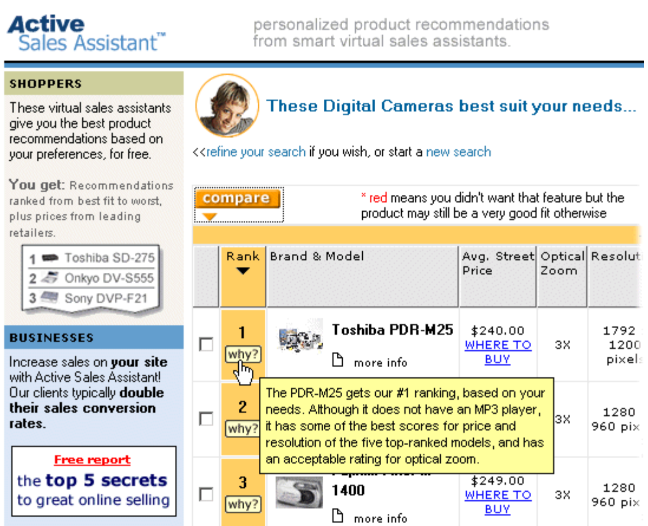
- Emergency Evacuation
- Robotics
- Space Exploration
- Manufacturing Analysis
- Games (e.g. Bridge)
- Product Recommendations
- …

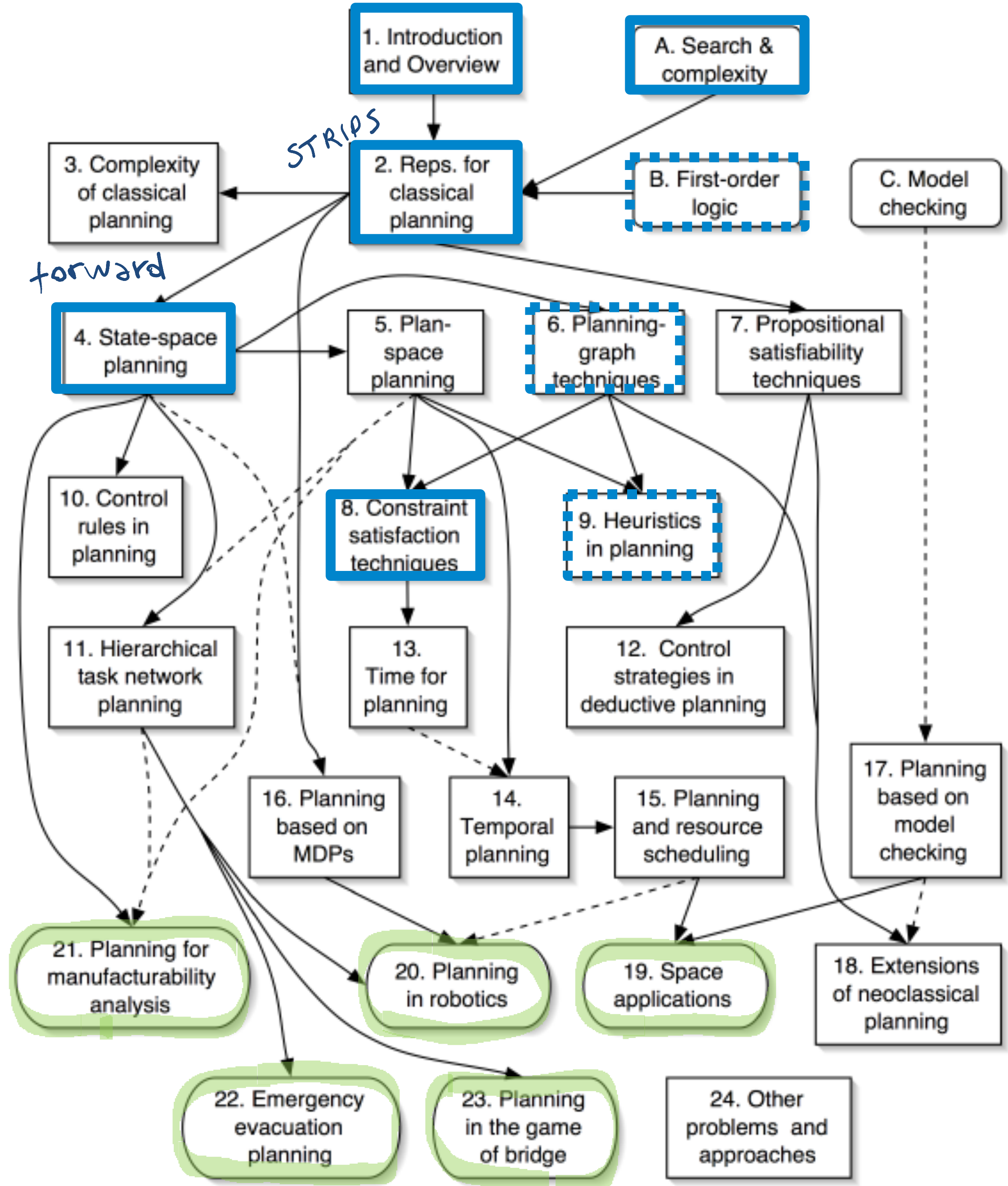









# You know the key ideas! ☺

– Ghallab, Nau, and Traverso ***Automated Planning: Theory and Practice***
Web site:
- http://www.laas.fr/planning
- Also has lecture notes

You know

You know a little

Applications

# Lecture Overview

- Recap: STRIPS, forward planning & heuristics
- Recap: CSP planning
- More CSP planning
  - Details on CSP representation
  - Solving the CSP planning problem
- Time permitting: Intro to Logic

# Course Overview

Course Module: *Representation* — Reasoning Technique

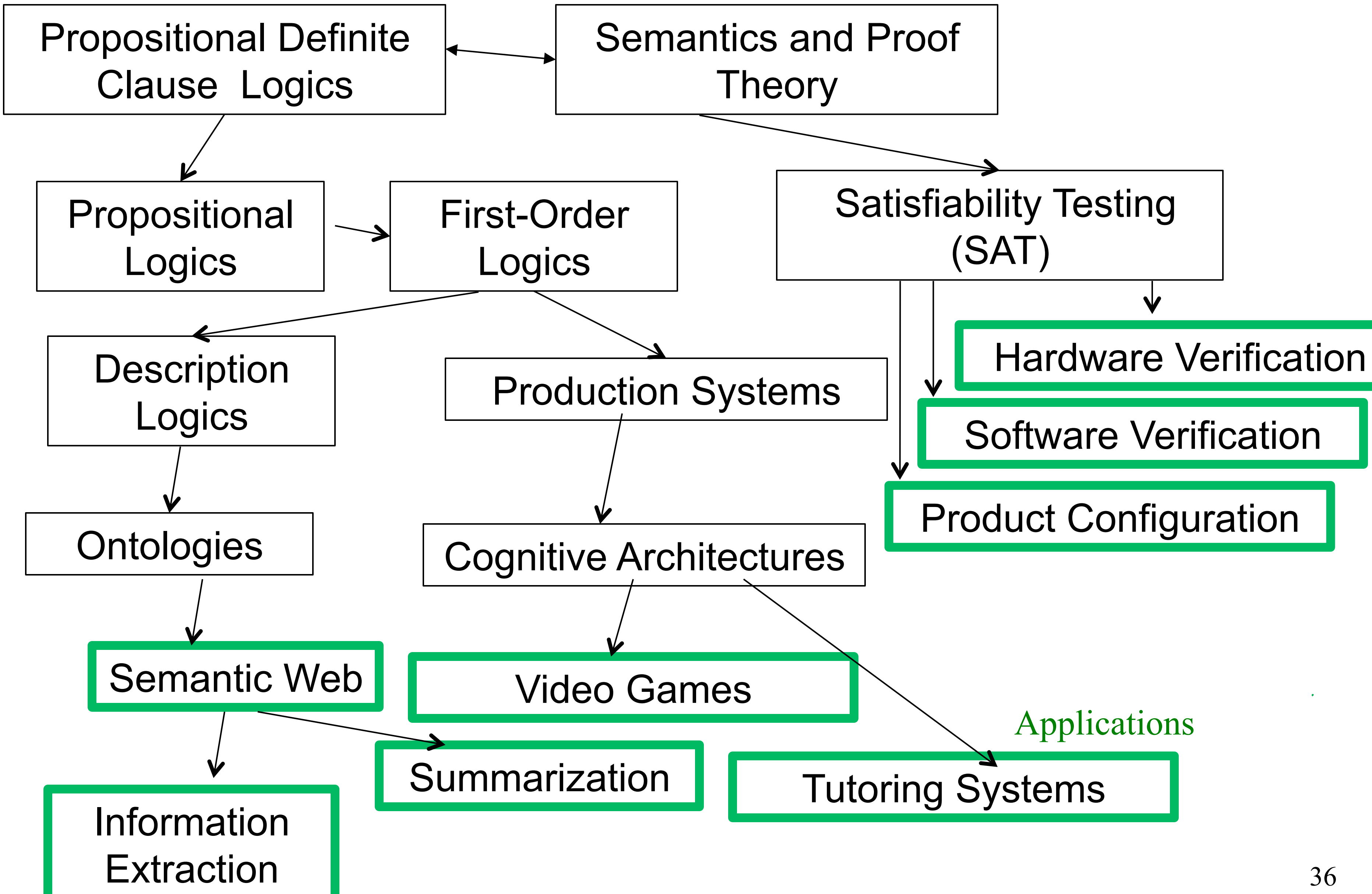
**Environment**

**Problem Type**

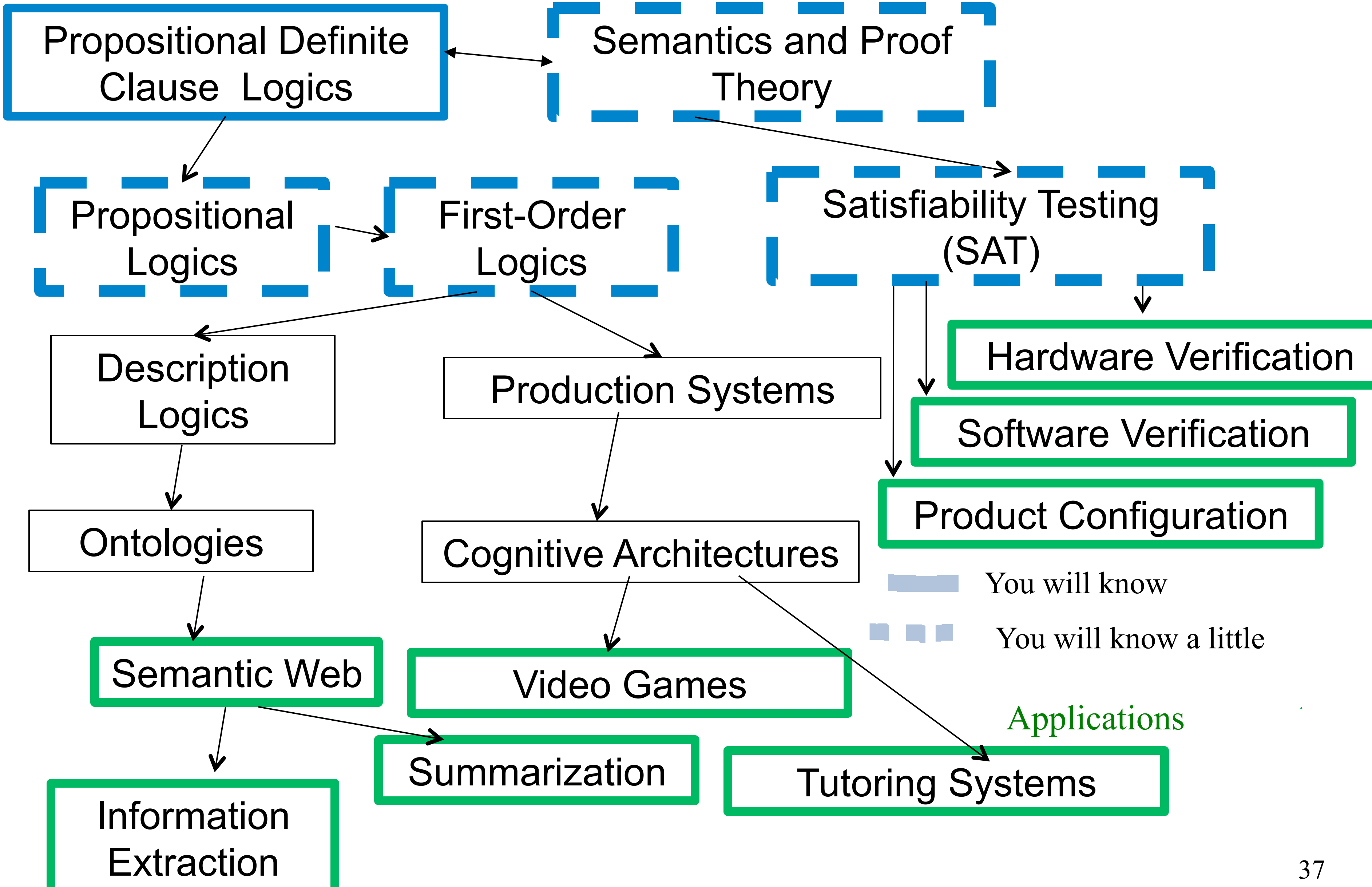
| Problem Type | | Deterministic | Stochastic | Course Module |
|---|---|---|---|---|
| Static | Constraint Satisfaction | ***Variables + Constraints*** — Arc Consistency; Search | | |
| Static | Logic | ***Logics*** — Search | ***Bayesian Networks*** — Variable Elimination | Uncertainty |
| Sequential | Planning | ***STRIPS*** — Search; As CSP (using arc consistency) | ***Decision Networks*** — Variable Elimination; ***Markov Processes*** — Value Iteration | Decision Theory |

Back to static problems, but with richer representation

# Logics in AI: Similar slide to the one for planning



# Logics in AI: Similar slide to the one for planning



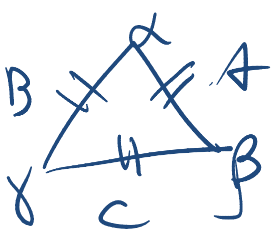
# What you already know about logic...

- **From programming: Some logical operators**

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

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 prove statements about those




$$\forall(x)\,triangle(x) \longrightarrow [A = B = C \longleftrightarrow \alpha = \beta = \gamma]$$

We use logic as a Representation and Reasoning System that can be used to formalize a domain and to reason about it

# Logic: a framework for representation & reasoning

- When we represent a domain about which we have only partial (but certain) information, what are some of the things we need to represent?

# Logic: a framework for representation & reasoning

- When we represent a domain about which we have only partial (but certain) information, we need to represent….
  - Objects, properties, sets, groups, actions, events, time, space, …
- All these can be represented as
  - Objects
  - Relationships between objects
- Logic is the language to express knowledge about the world this way
- http://en.wikipedia.org/wiki/John_McCarthy (1927 - 2011)

  Logic and AI "The Advice Taker"

  Coined "Artificial Intelligence". Dartmouth W'shop (1956)

# Why Logics?

- "**Natural**" to express **knowledge** about the world
- (more natural than a "flat" set of variables & constraints)

- *e.g. "Every 101 student will pass the course"*

  *Course (c1)*

  *Name-of (c1, 101)*

  $$\forall(z) student(z) \,\&\, registered(z,c1) \longrightarrow will\_pass(z,c1)$$

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

# Learning Goals for Planning

- STRIPS
  - Represent a planning problem with the STRIPS representation
  - Explain the STRIPS assumption
- Forward planning
  - 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
- CSP planning
  - Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
  - Solve a planning problem with CSP by expanding the horizon

---

- Assignment 3 is available
  - Due in 2 weeks. Do the planning part early
- Useful to do practice exercise 7 before the assignment