Planning: Representation

CPSC 322 – Planning 1

Textbook §8.0-8.2

February 21, 2011

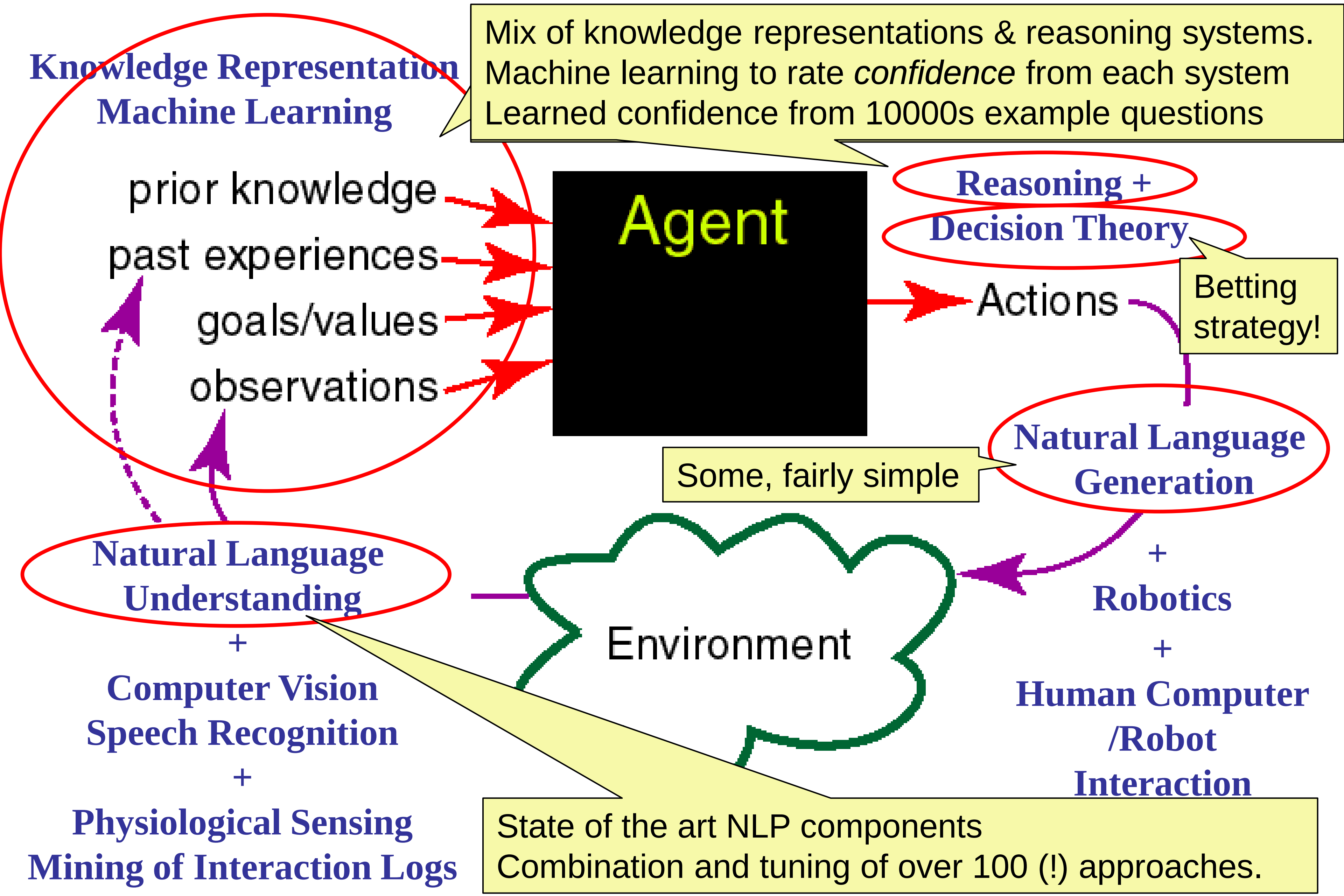
Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting: forward planning (planning as search)

IBM Watson

- Very impressive performance
  - Clearly won against the two most accomplished Jeopardy experts
- Solves a very complex problem: question answering
  - Much harder for AI than logical problems like chess or proofs
  - Dealing with uncertainty $\rightarrow$ last 2 modules in the course + 422
- Knowledge of its own confidence is particularly important
- Many potential applications
  - Medicine
  - Business
  - …

Watson as an intelligent agent (see lecture 1)



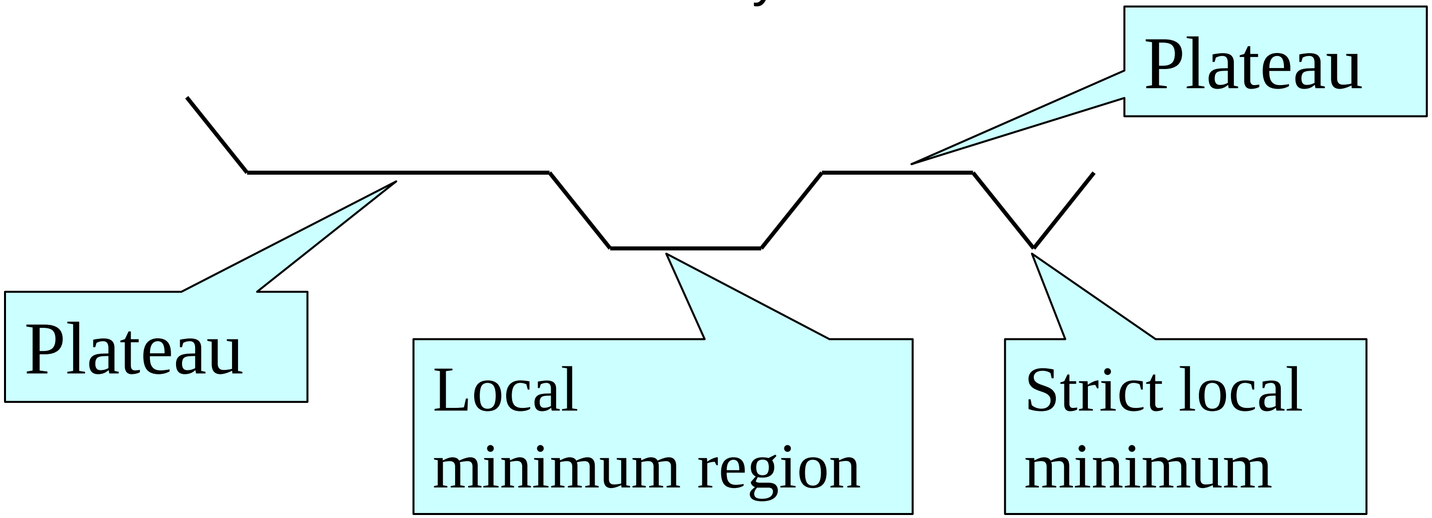
Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting:
  start of forward planning (planning as search)

P.S. Definition of a plateau

- Local minimum
  - Search state n such that all its neighbours n’ have h(n’) > h(n)

- Plateau
  - Set of connected states $\{n_1, \ldots, n_k\}$ with $h(n_1) = h(n_2) = \ldots = h(n_k)$
    - At least one of the n_i has a neighbour n’ with $h(n') < h(n_i)$
  - Problem: some problem instances have very large plateaus, need to search them effectively



Types of SLS algorithms

- Simulated Annealing
- Tabu Search
- Iterated Local Search
- (Stochastic) Beam Search
- Genetic Algorithms

- Only need to know high-level concepts

How to set the parameters?

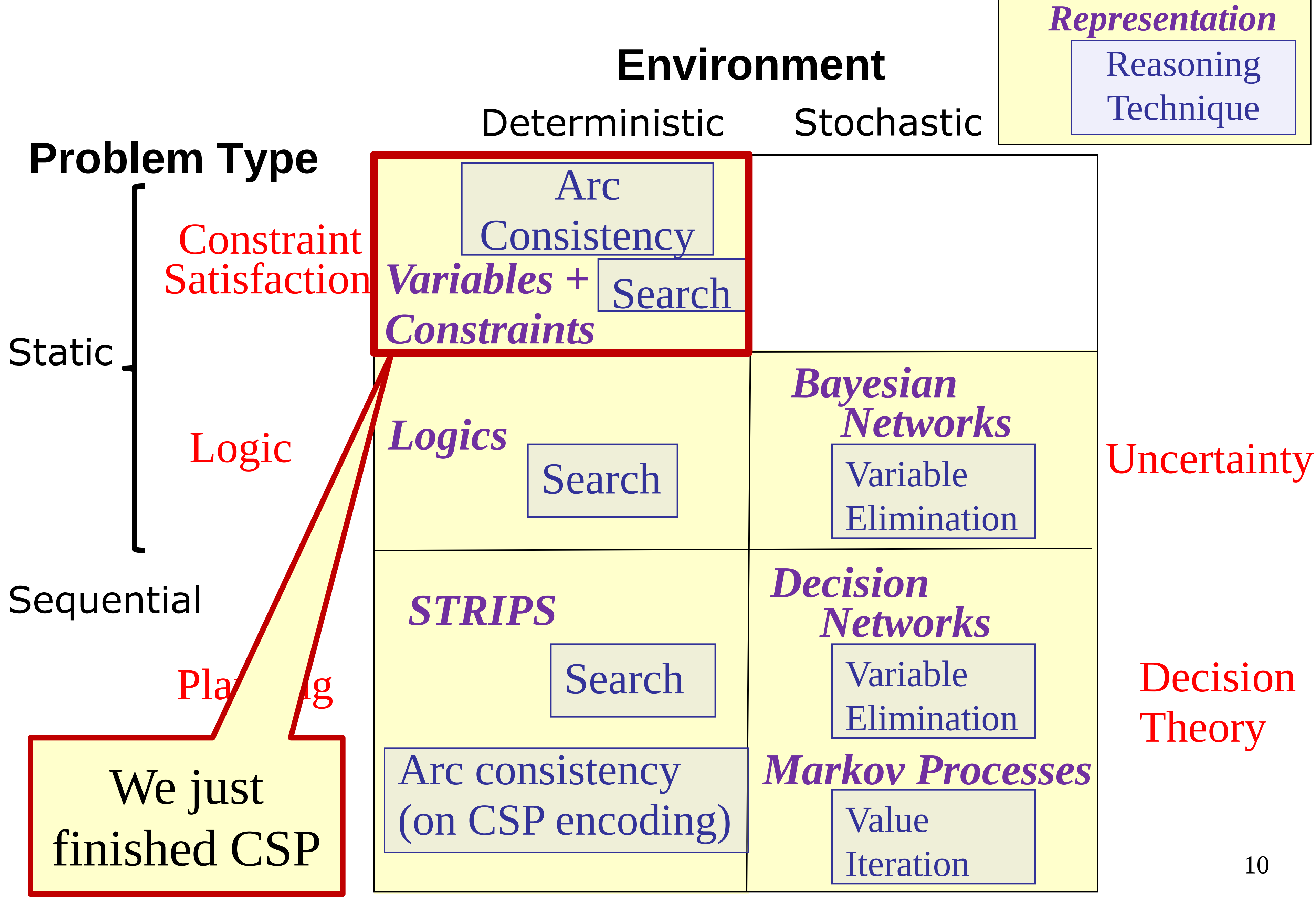
- “Automated algorithm configuration”
  - Optimize the performance of arbitrary parameterized algorithms

- “Parameter” is a very general concept
  - Numerical domains: real or integer
  - Categorical domains: finite and unordered
    - Alternative heuristics to use in A*
    - Alternative data structures
    - Alternative Java classes in a framework implementation
    - …

Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting:
  start of forward planning (planning as search)

Course Overview



Course Overview

Course Module: ***Representation*** / Reasoning Technique

Environment

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

Now we start planning

Planning

- With CSPs, we looked for solutions to essentially atemporal problems
  - find a single variable assignment (state) that satisfies all of our constraints
  - did not care about the path leading to that state

- Now consider a problem where we are given:
  - A description of an initial state
  - A description of the effects and preconditions of actions
  - A goal to achieve

- ...and want to find a sequence of actions that is possible and will result in a state satisfying the goal
  - note: here we want not a single state that satisfies our constraints, but rather a sequence of states that gets us to a goal

Key Idea of Planning

- Open up the representation of states, goals and actions
  - States and goals as features (variable assignments), like in CSP
  - Actions as preconditions and effects defined on features

- Agent can reason more deliberately about what actions to consider to achieve its goals.

Contrast this to simple graph search

- How did we represent the problem in graph search?
  - States, start states, goal states, and successor function
    - Successor function: when applying action a in state s, you end up in s'
- We used a "flat" state-based representation
  - there's no sense in which we can say that states a and b are more similar than states a and z (they're just nodes in a graph)
  - Thus, we can't represent the successor function any more compactly than a tabular representation

Starting state	Action	Resulting state
⋮	⋮	⋮

Problems with the Tabular Representation

- Usually too many states for a tabular representation to be feasible

- Small changes to the model can mean big changes for the representation
  - e.g., if we added another variable, all the states would change

- There may be structure and regularity
  - to the states
  - and to the actions
  - no way to capture this with a tabular representation

Feature-Based Representation

- Features helped us to represent CSPs more compactly than states could
  - The main idea: factor states into joint variable assignments
  - Each constraint only needed to mention the variables it constrains
  - That enabled efficient constraint propagation: arc consistency
  - No way to do this in flat state-based representation

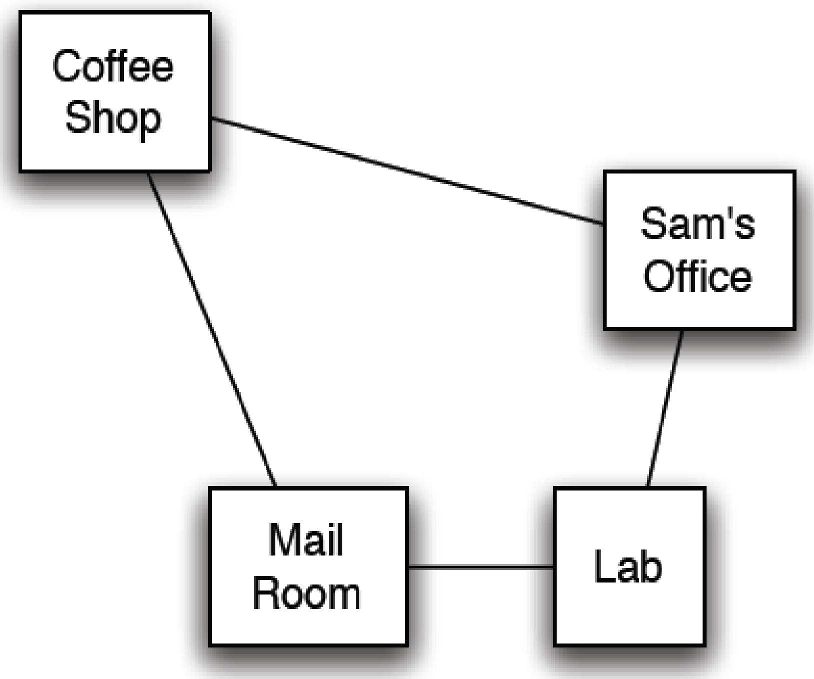
- Want to use similar idea when searching for a sequence of actions that brings us from a start state to a goal stated
  - Main idea: compact, rich representation and efficient reasoning

Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting:
  start of forward planning (planning as search)

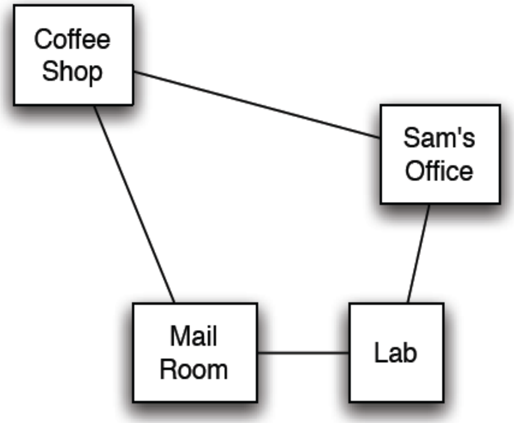
Delivery Robot Example (textbook)

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



Delivery Robot Example: features

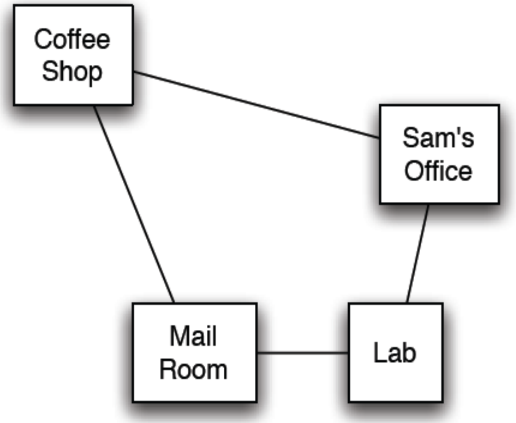
- RLoc - Rob's location
  - Domain: {coffee shop, Sam's office, mail room, laboratory} short {cs, off, mr, lab}
- RHC – Rob has coffee
  - Domain: {true, false}. By rhc indicate that Rob has coffee, and by $\overline{rhc}$ that Rob doesn't have coffee
- SWC – Sam wants coffee {true, false}
- MW – Mail is waiting {true, false}
- RHM – Rob has mail {true, false}



- An example state is $\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
- How many states are there?

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)

PUC - Rob picks up coffee

- must be at the coffee shop

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

Preconditions for action application

Example State-Based Representation

State	Action	Resulting State
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mc \rangle$	$\langle mr, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle lab, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle mac \rangle$	$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$
$\langle off, \overline{rhc}, swc, \overline{mw}, rhm \rangle$	$\langle dm \rangle$	$\langle off, \overline{rhc}, \overline{swc}, \overline{mw}, \overline{rhm} \rangle$
$\vdots$	$\vdots$	$\vdots$

Tabular representation:

need an entry for every state and every action applicable in that state!

Example for more compact representation

- A representation of the action pick up coffee, PUC:
  - Only changes a subset of features
    - In this case, only RHC (Rob has coffee)
  - Only depends on a subset of features
    - In this case, Loc = cs (Rob is in the coffee shop)

  – preconditions Loc = cs and RHC = $\overline{rhc}$
  – effects RHC = rhc

Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting:
  start of forward planning (planning as search)

Feature-Based Representation

- Where we stand so far:
  - the state-based representation is unworkable
  - a feature-based representation might help

- How would a feature-based representation work?
  - states are easy, just as in CSP: joint assignment to variables
    - Includes initial states and goal states
  - the key is modeling actions

Modeling actions

- To “model actions" in the feature-based representation, we need to solve two problems:
  – Model when the actions are possible, in terms of the values of the features of the current state
  – Model the state transitions in a “factored" way

- Why might this be more tractable/manageable than the tabular representation?
  – If actions only depend on/modify some features
    - Representation will be more compact (exponentially so!)
    - The representation can be easier to modify/update

The STRIPS Representation

- For reference:
  The book also discusses a feature-centric representation
  - for every feature, where does its value come from?
  - causal rule: ways a feature's value can be changed by taking an action.
  - frame rule: requires that a feature's value is unchanged if no action changes it.
- STRIPS is an action-centric representation:
  - for every action, what does it do?
- This leaves us with no way to state frame rules.
- The STRIPS assumption:
  - all variables not explicitly changed by an action stay unchanged

STRIPS representation

(STanford Research Institute Problem Solver)

In STRIPS, an action has two parts:

1. Preconditions: a set of assignments to variables that must be satisfied in order for the action to be legal

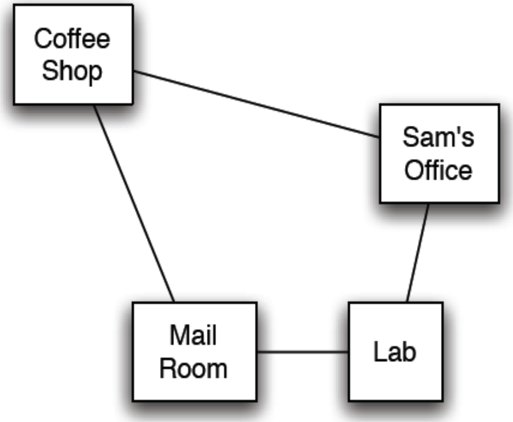
2. Effects: a set of assignments to variables that are caused by the action

Example

- STRIPS representation of the action pick up coffee, PUC:
  - preconditions Loc = cs and RHC $\overline{rhc}$
  - effects RHC = rhc

- STRIPS representation of the action deliver coffee, DelC:
  - preconditions Loc = off and RHC = rhc
  - effects RHC = $\overline{rhc}$ and SWC = $\overline{swc}$

- Note that Sam doesn't have to want coffee for Rob to deliver it; one way or another, Sam doesn't want coffee after delivery.

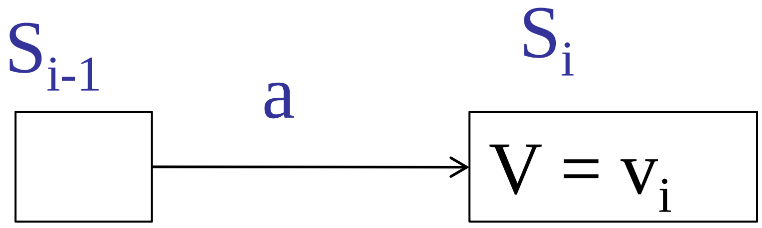


STRIPS (cont')

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?



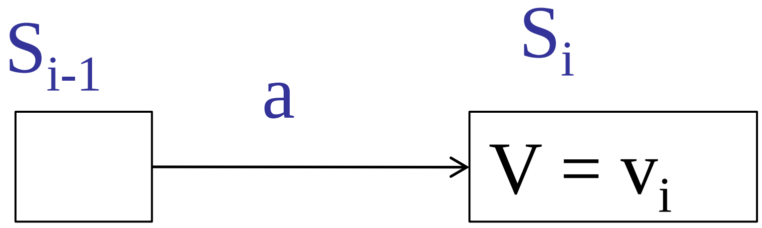
What can we conclude about a and/or the state of the world S_{i-1} ,immediately preceding the execution of a?

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

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

At least one of the above

Both of the above



Lecture Overview

- IBM Watson
- Recap: types of SLS algorithms
- Planning: intro
- Planning: example
- STRIPS: A Feature-Based Representation
- Time-permitting:
  start of forward planning (planning as search)

Solving planning problems

- STRIPS lends itself to solve planning problems either

  - As pure search problems
  - As CSP problems

- We will look at one technique for each approach

Forward planning

- Idea: search in the state-space graph
  - The nodes represent the states
  - The arcs correspond to the actions:
    - The arcs from a state s represent all of the actions that are possible in state s
  - A plan is a path from the state representing the initial state to a state that satisfies the goal

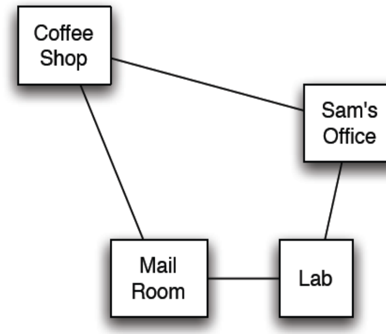
- What actions a are possible in a state s?

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

Example state-space graph: first level



Actions

mc: move clockwise

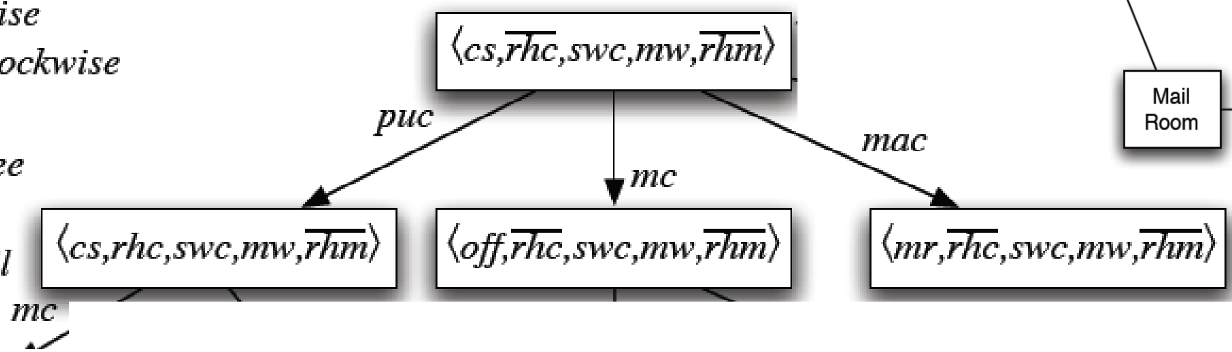
mac: move anticlockwise

puc: pick up coffee

dc: deliver coffee

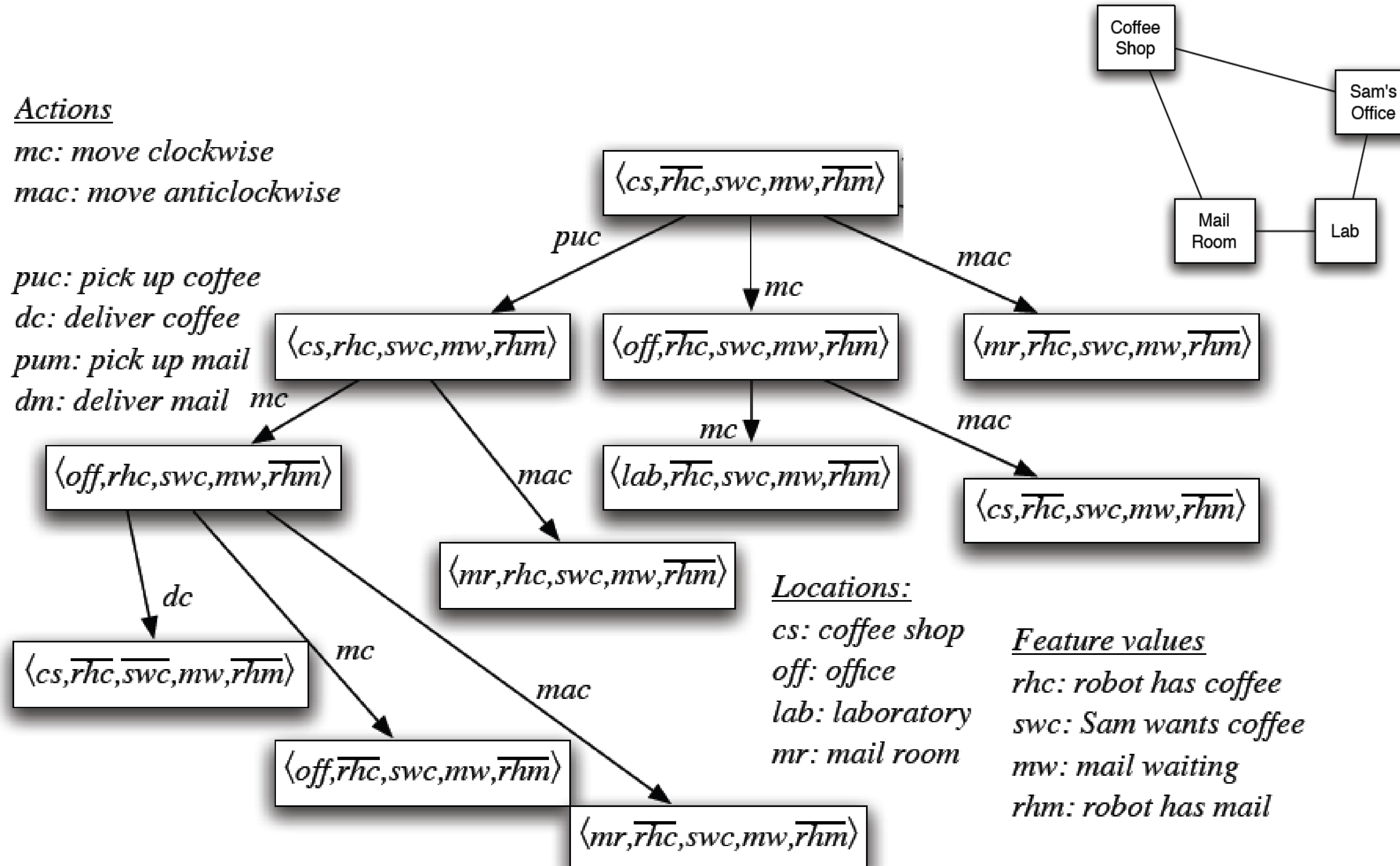
pum: pick up mail

dm: deliver mail



Goal: $\overline{swc}$

Part of state-space graph



Actions

mc: move clockwise
mac: move anticlockwise

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

Goal: $\overline{swc}$

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: next time
- Inference
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function

Learning Goals for today's class

- 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.

- Coming up:
  - Assignment 2 due on Wednesday
    - Can only use 2 late days
  - Midterm next Monday: FSC 1005, 3-4:30pm
    - 60% short answer questions. See WebCT for samples.
    - 40% long answer questions. See WebCT for an example.
  - Extra office hours this week
    - After class in the classroom for an hour
    - Tuesday & Thursday 3pm-4pm