Constraint Satisfaction Problems (CSPs)

Introduction

Computer Science cpsc322, Lecture 11

(Textbook Chpt 4.0 – 4.2)

January, 28, 2009

Announcements

- Only one more week for assignment1

- Search wrap-up
  - Go back to learning goals (end of slides)
  - Make sure you understands the inked slides
  - More details or different examples on textbook
  - Work on the practice exercises
  - If still confused, come to office hours

Lecture Overview

- **Generic Search vs. Constraint Satisfaction Problems**
- Variables
- Constraints
- CSPs

Standard Search

To learn about **search** we have used it as the *reasoning strategy* for a **simple goal-driven planning agent**, but…

vacuum-cleaner Delivery Robot

8-puzzle Marine-Problem

SOLUTION: path from start state to goal

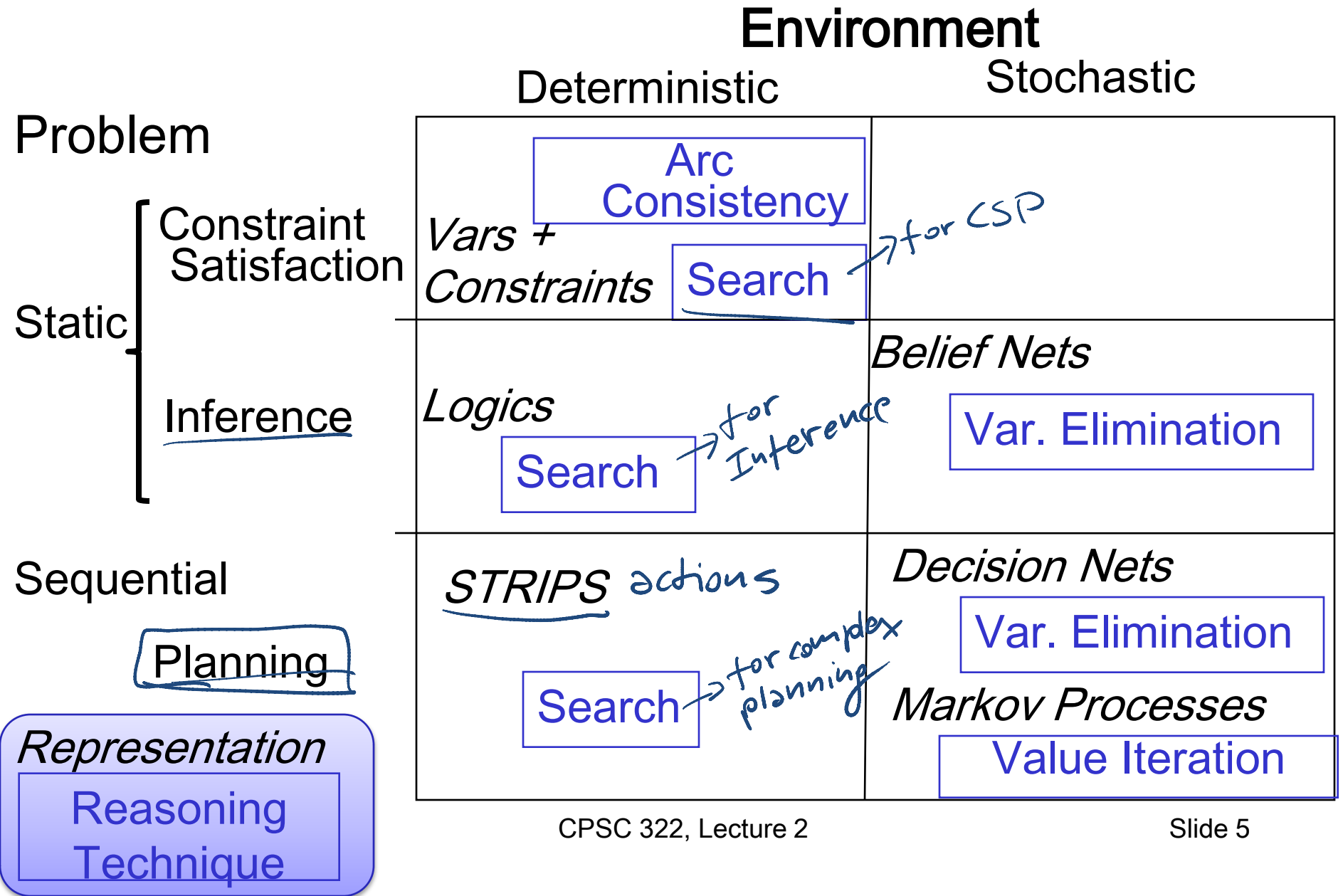
Standard search problem: An agent can solve a problem by searching in a space of states

- state is a "black box" – any arbitrary data structure that supports three problem-specific routines

goal(n) heuristic(n)

successor/neighbor(n)

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



Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

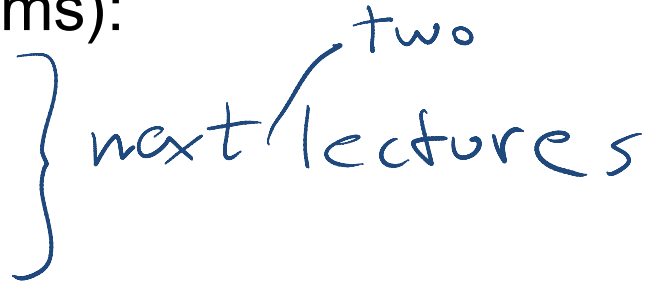
- State
- Successor function
- Goal test
- Solution

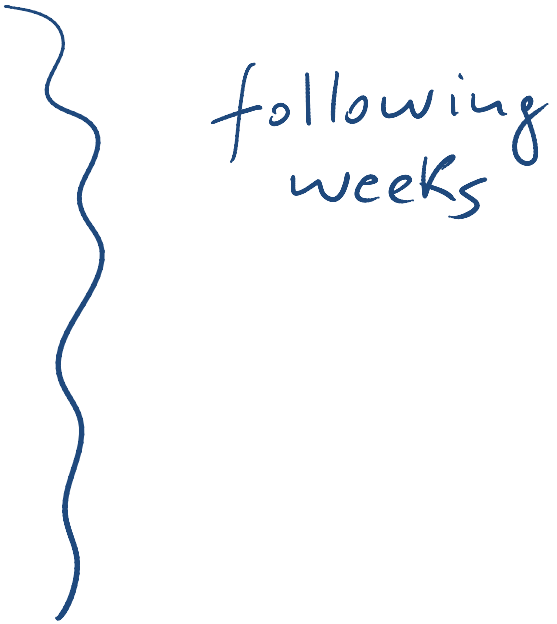
Planning :

- State
- Successor function
- Goal test
- Solution

Inference

- State
- Successor function
- Goal test
- Solution





Lecture Overview

- Generic Search vs. Constraint Satisfaction Problems
- Variables/Features
- Constraints
- CSPs

Variables/Features, domains and Possible Worlds

- Variables / features
  - we denote variables using capital letters A, B
  - each variable V has a domain *dom*(V) of possible values

$dom(A) = dom(B) = \{0, 1\}$

- Variables can be of several main kinds:
  - Boolean: |*dom*(V)| = 2 called propositions
  - Finite: the domain contains a finite number of values
  - ~~Infinite but Discrete: the domain is countably infinite~~ not in this course
  - ~~Continuous: e.g., real numbers between 0 and 1~~

- Possible world: a complete assignment of values to a set of variables

e.g. $\{A = 0, B = 1\}$

Possible Worlds

Mars Explorer Example

- Weather {S, C} (sunny, cloudy)
- Temperature {-40, ..., +40}
- LocX 0° ... 359° (longitude)
- LocY 0° ... 179° (latitude)

One possible state {S, +35, 30°, 110°}

2 * 81 * 360 * 180

number of possible worlds

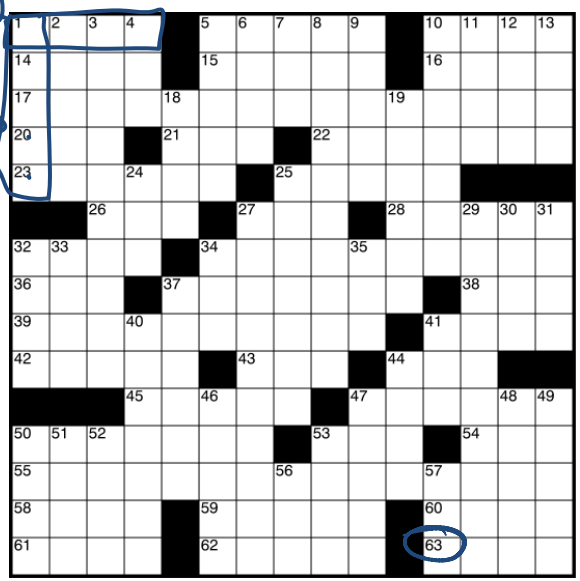
mutually exclusive

product of cardinality of each domain (i.e. number of possible values)

So it is always exponential in the number of variables!

Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are valid English words
  - possible worlds: all ways of assigning words



all words of length 4

word

v_1

valid

of required length

$\sim 150 * 10^3$ number of English words

"assuming" same number of words for each length from 1 to 10

we have $\not\sim 150 * 10^2$ words of length 1

So # of possible world $> (150 * 10^2)^{63}$

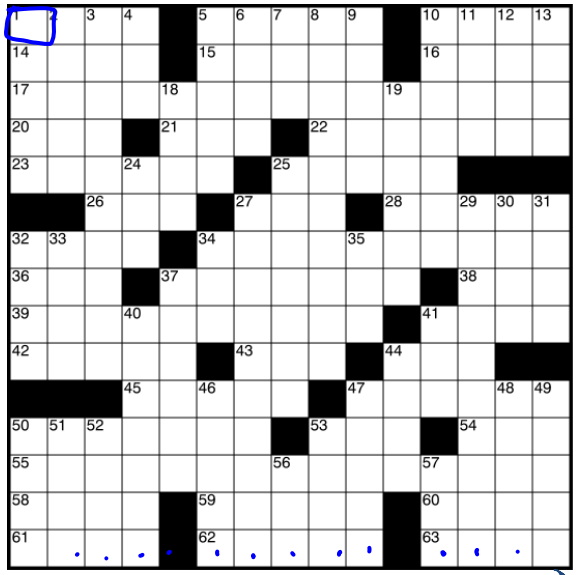
63 ← approx. # of vars

↑ cardinality of dom of each var

More Examples

- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - possible worlds: all ways of assigning letters to cells

# of possible worlds ~ 26



- Sudoku:
  - variables are cells
  - domains are numbers between 1 and 9
  - possible worlds: all ways of assigning numbers to cells

# of possible worlds = 9

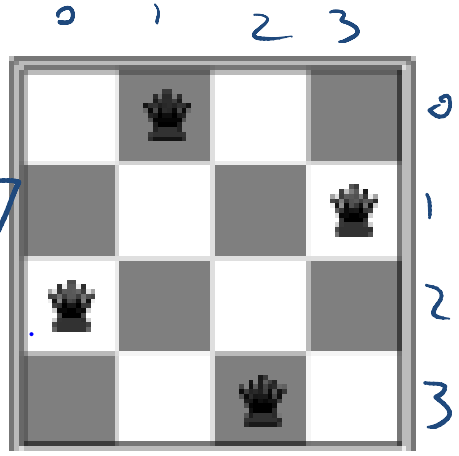
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

More examples

- n-Queens problem
  - variable: location of a queen on a chess board
    - there are *n* of them in total, hence the name
  - domains: grid coordinates
  - possible worlds: locations of all queens

$$\binom{n^2}{n} = \frac{n^2!}{(n^2-n)!\,n!} \qquad \frac{16!}{12!\,4!}$$

possible ways to choose n location out of n^2



16 Possible locations:

(0,0), (0,1) . . . . . (3,3)

More examples

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - possible worlds: time/location assignments for each task

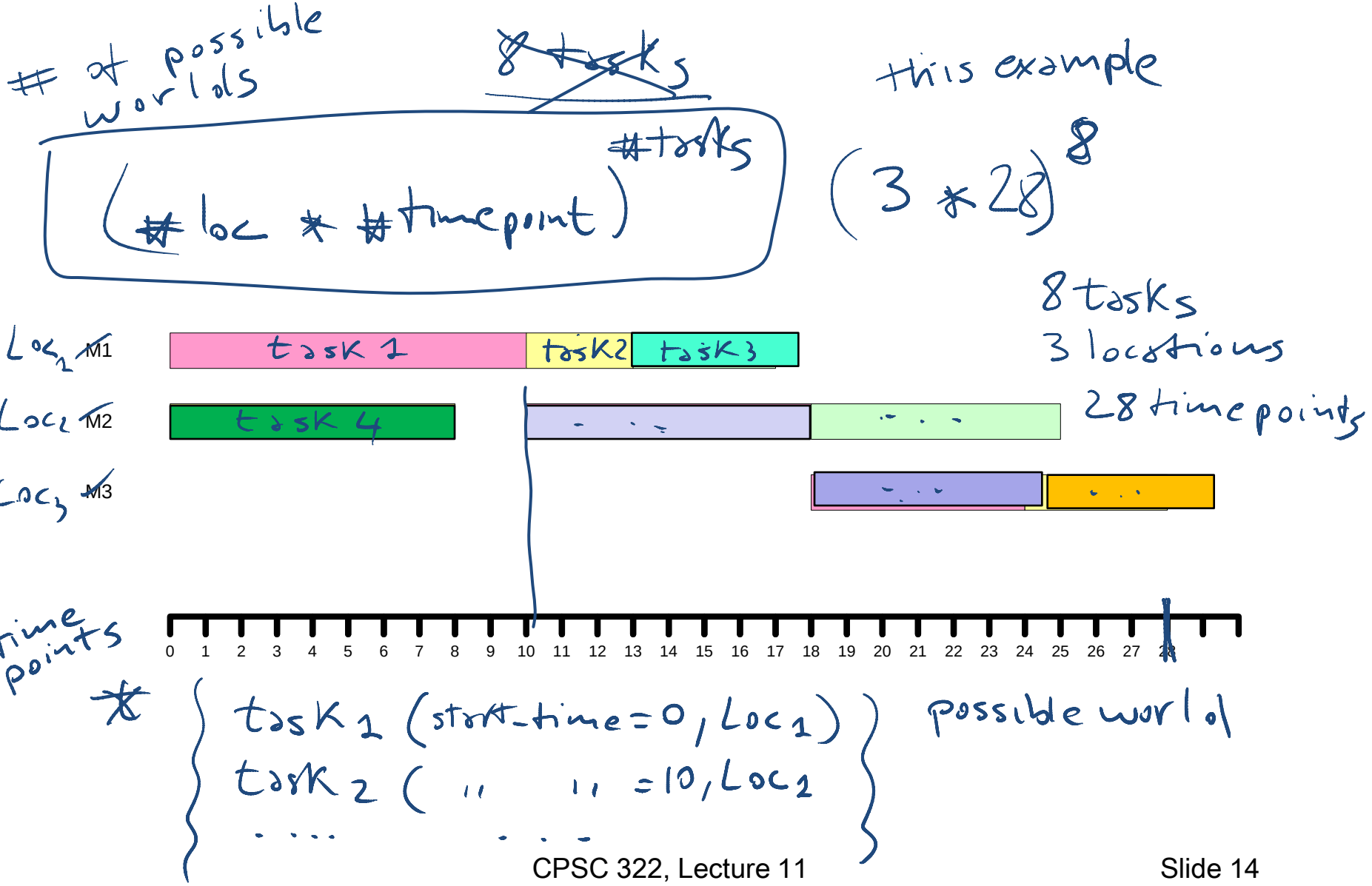
task$_1$ task$_2$ ...

assume time length of task is fixed, you only need to specify start-time

(start_time, location)

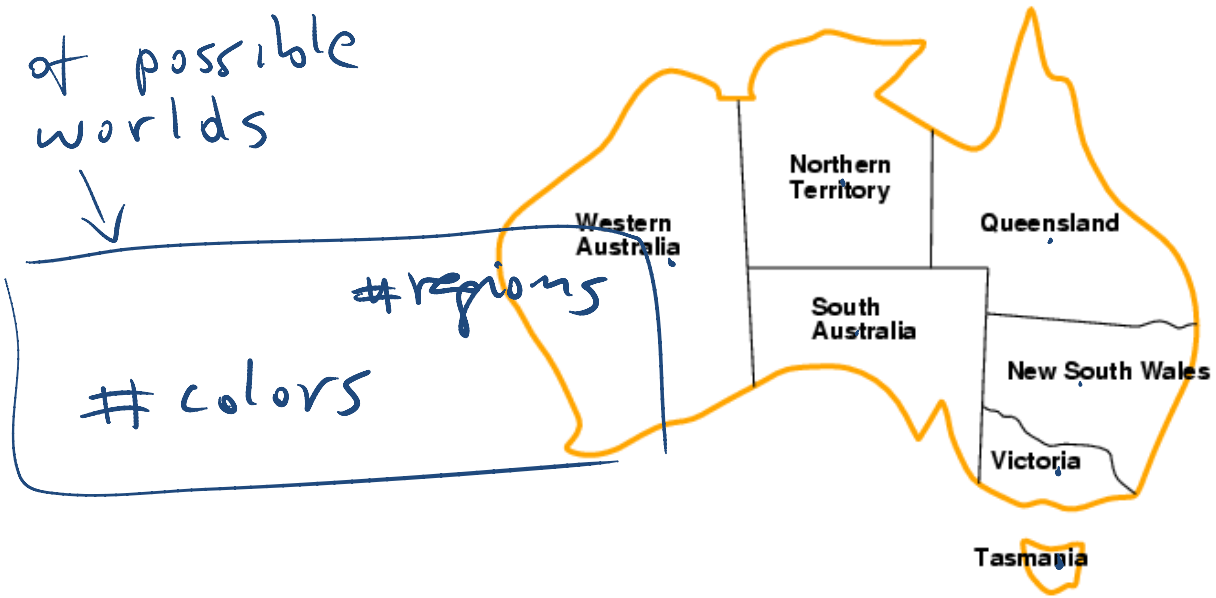
e.g. task$_1$ = {11am.., room 310}

Scheduling possible world



More examples….

- Map Coloring Problem
  - variable: regions on the map
  - domains: possible colors
  - possible worlds: color assignments for each region



Lecture Overview

- Generic Search vs. Constraint Satisfaction Problems
- Variables/Features
- Constraints
- CSPs

Constraints

Constraints are restrictions on the values that one or more variables can take

- Unary constraint: restriction involving a single variable

  $A = 0$ $A, B, C \{0, 1\}$ same domain

- k-ary constraint: restriction involving the domains of k different variables $A > B$ $A > B + C$
  - it turns out that k-ary constraints can always be represented as binary constraints, so we'll *probably* only talk about this case

- **Constraints can be specified by**
  - giving a list of valid domain values for each variable participating in the constraint $\{A=0, B=0\}$ $\{A=1, B=1\}$
  - giving a function that returns true when given values for each variable which satisfy the constraint $A = B$

Example: Map-Coloring

Variables *WA, NT, Q, NSW, V, SA, T*

Domains D_i = {red,green,blue}

Constraints: adjacent regions must have different colors

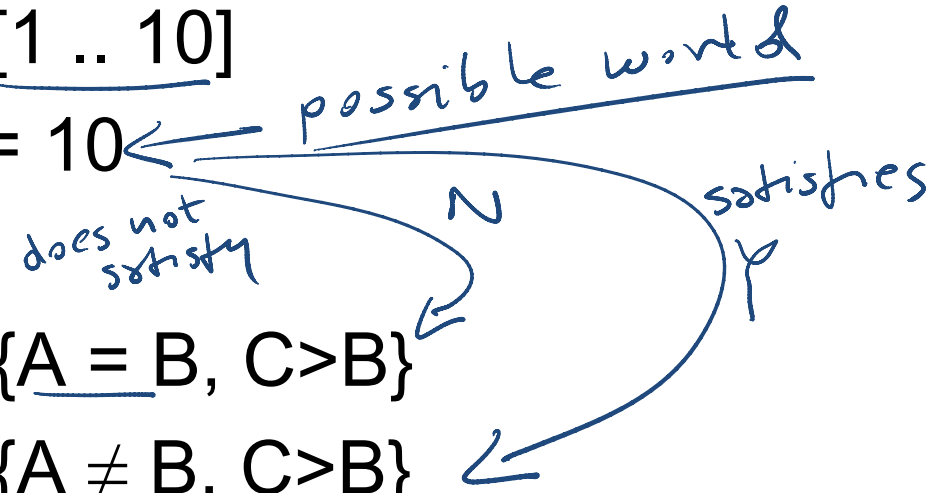
e.g., (WA,NT) in {(red,green),(red,blue),(green,red),
(green,blue),(blue,red),(blue,green)}

or WA ≠ NT,

Constraints (cont.)

- A **possible world** satisfies **a set of constraints** if the set of variables involved in each constraint take values that are consistent with that constraint

- A,B,C domains [1 .. 10]
- A= 1 , B = 2, C = 10
- Constraint set1 {A = B, C>B}
- Constraint set2 {A ≠ B, C>B}

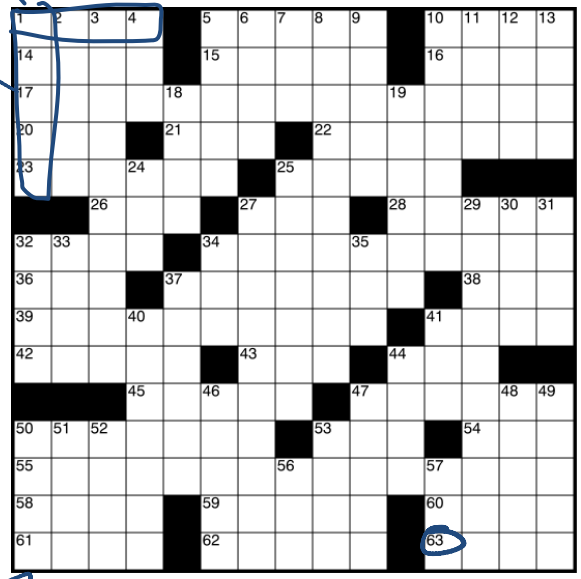


Examples

$h_1[0] = v_1[0]$

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are valid English words
  - *constraints:* words have the same letters at points where they intersect

~ 225 constraints



- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - *constraints:* sequences of letters form valid English words

~63 constraints

conc (A[0,0] . . . . . A[0,3]) ∈ 4 letter English word

Examples

- Sudoku:
  - variables are cells
  - domains are numbers between 1 and 9
  - *constraints:* rows, columns, boxes contain all different numbers

# of constraint = # empty-cells * 24

$A[0,2] \neq A[0,1]$; $A[0,2] \neq A[0,0]$ ..... solution

A

	0	1	2						
0	5	3			7				
1	6			1	9	5			
2		9	8					6	
	8				6				3
	4			8		3			1
	7				2				6
		6					2	8	
				4	1	9			5
					8			7	9

for all i $i \neq 2$

$A[2,0] \neq A[i,0]$ which means

$A[2,0] \neq A[0,0]$

$A[2,0] \neq A[1,0]$

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

More examples

- n-Queens problem
  - variable: location of a queen on a chess board
    - there are *n* of them in total, hence the name
  - domains: grid coordinates
  - *constraints*: no queen can attack another

eg two queens cannot be on the same column / row

$Q_1 = \{x_1, y_1\}$

$Q_2 = \{x_2, y_2\}$

$x_1 = x_2$ and $y_1 = y_2$

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - *constraints*: e.g. $task_1(loc_1, start\text{-}t_1)$ if $start\text{-}t_1 = start\text{-}t_2$ $task_2(loc_2, start\text{-}t_2)$ then $loc_1 \neq loc_2$
    - ✓ tasks can't be scheduled in the same location at the same time;
    - ✓ certain tasks can be scheduled only in certain locations;
    - ✓ some tasks must come earlier than others; etc.

Lecture Overview

- Generic Search vs. Constraint Satisfaction Problems
- Variables/Features
- Constraints
- CSPs

Constraint Satisfaction Problems: definitions

Definition (Constraint Satisfaction Problem)

A constraint satisfaction problem consists of

- a set of variables
- a domain for each variable
- a set of constraints

possible worlds

Definition (model / solution)

A model of a CSP is an assignment of values to variables that satisfies all of the constraints.

possible world

Example: Map-Coloring

Variables *WA, NT, Q, NSW, V, SA, T*

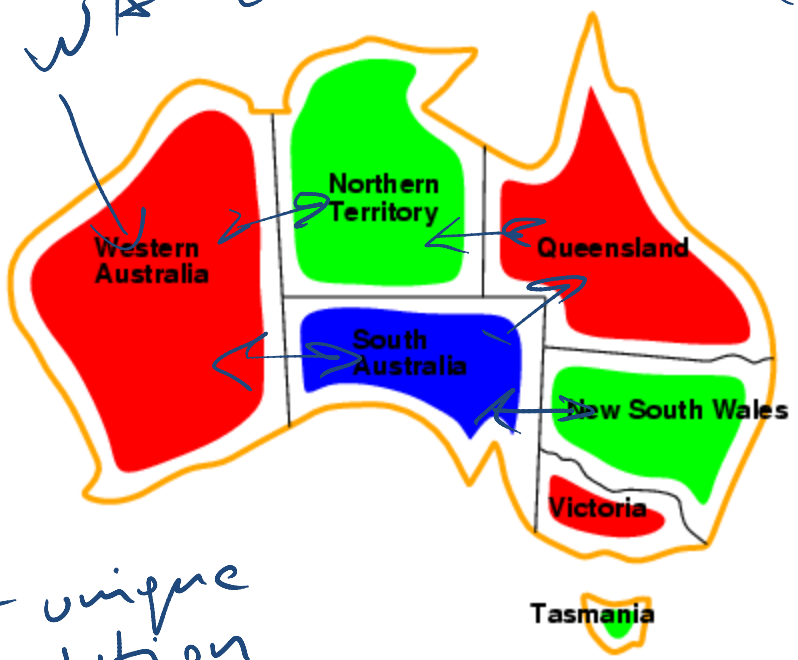
Domains D_i = {red,green,blue}

Constraints: adjacent regions must have different colors

e.g., WA ≠ NT, or

(WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),(blue,red),(blue,green)}

Example: Map-Coloring



Models / Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red,SA = blue, T = green

Constraint Satisfaction Problem: Variants

We may want to solve the following problems using a CSP

- determine whether or not a model exists
- find a model
- find all of the models
- count the number of the models
- find the best model given some model quality
  - this is now an optimization problem
- determine whether some properties of the variables hold in all models

To summarize

- Need to think of search beyond simple goal driven planning agent.
- We started exploring the first AI Representation and Reasoning framework: CSPs

Next class

CSPs: Search and Arc Consistency

(Textbook Chpt 4.3-4.5)

Learning Goals for today's class

- Define possible worlds in term of variables and their domains.
- Compute number of possible worlds on real examples
- Specify constraints to represent real world problems differentiating between:
  - Unary and k-ary constraints
  - List vs. function format.

Verify whether a possible world satisfies a set of constraints (i.e., whether it is a model, a solution)

Extra slide (may be used here?)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Goal state: 9×9 grid completely filled so that

- each column,
- each row, and
- each of the nine 3×3 boxes
- contains the digits from 1 to 9, only *one* time each

A possible **start state** (partially completed grid)

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9