Solving Constraint Satisfaction Problems (CSPs) using Search

CPSC 322 – CSP 2

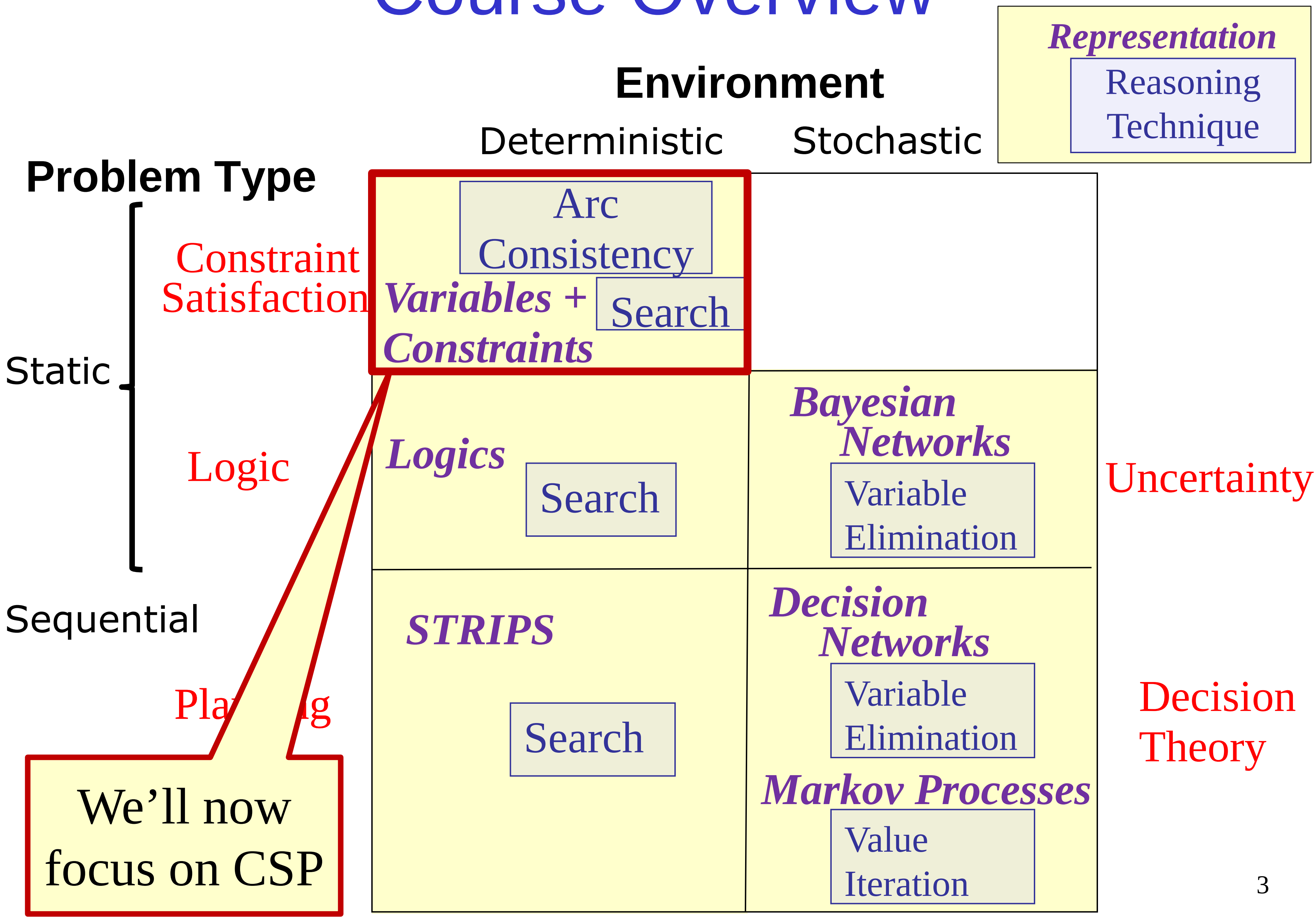
Textbook § 4.3-4.4

January 31, 2011

Lecture Overview

- Constraint Satisfaction Problems (CSPs): Definition and Recap
- Constraint Satisfaction Problems (CSPs): Motivation
- Solving Constraint Satisfaction Problems (CSPs)
  - Generate & Test
  - Graph search
  - Arc consistency (start)

Course Overview



Standard Search vs. CSP

- First studied general state space search in isolation
  - Standard search problem: search in a state space

- State is a “black box” : any arbitrary data structure that supports three problem-specific routines
  - goal test: goal(state)
  - finding successor nodes: neighbors(state)
  - if applicable, heuristic evaluation function: h(state)

- We’ll see more specialized versions of search for various problems

Search in Specific R&R Systems

- Constraint Satisfaction Problems:
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function
- Planning :
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function
- Inference
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function

Constraint Satisfaction Problems (CSPs): Definition

Definition:

A constraint satisfaction problem (CSP) consists of:

- a set of variables $\mathcal{V}$
- a domain dom(V) for each variable V $\in \mathcal{V}$
- a set of constraints $\mathcal{C}$

Simple example:

- $\mathcal{V} = \{V_1\}$
  - $dom(V_1) = \{1,2,3,4\}$
- $\mathcal{C} = \{C_1,C_2\}$
  - C_1: $V_1 \neq 2$
  - C_2: $V_1 > 1$

Another example:

- $\mathcal{V} = \{V_1,V_2\}$
  - $dom(V_1) = \{1,2,3\}$
  - $dom(V_2) = \{1,2\}$
- $\mathcal{C} = \{C_1,C_2,C_3\}$
  - C_1: $V_2 \neq 2$
  - C_2: $V_1 + V_2 < 5$
  - C_3: $V_1 > V_2$

Constraint Satisfaction Problems (CSPs): Definition

Definition:
A constraint satisfaction problem (CSP) consists of:
- a set of variables $\mathcal{V}$
- a domain dom(V) for each variable $V \in \mathcal{V}$
- a set of constraints $\mathcal{C}$

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

Simple example:
- $\mathcal{V} = \{V_1\}$
  - $dom(V_1) = \{1,2,3,4\}$
- $\mathcal{C} = \{C_1, C_2\}$
  - C_1: $V_1 \neq 2$
  - C_2: $V_1 > 1$

All models for this CSP:

$\{V_1 = 3\}$

$\{V_1 = 4\}$

Constraint Satisfaction Problems (CSPs): Definition

Definition:

A constraint satisfaction problem (CSP) consists of:

- a set of variables $\mathcal{V}$
- a domain dom(V) for each variable $V \in \mathcal{V}$
- a set of constraints C

Definition:

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

Another example:

- $\mathcal{V} = \{V_1, V_2\}$
  - $dom(V_1) = \{1,2,3\}$
  - $dom(V_2) = \{1,2\}$
- $C = \{C_1, C_2, C_3\}$
  - C_1: $V_2 \neq 2$
  - C_2: $V_1 + V_2 < 5$
  - C_3: $V_1 > V_2$

Which are models for this CSP?

$\{V_1=1, V_2=1\}$

$\{V_1=2, V_2=1\}$

$\{V_1=3, V_2=1\}$

$\{V_1=3, V_2=2\}$

Possible Worlds

Definition:
A possible world of a CSP is an assignment of values to all of its variables.

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

I.e., a model is a possible world that satisfies all constraints

Another example:

- $\mathcal{V} = \{V_1,V_2\}$
  - $dom(V_1) = \{1,2,3\}$
  - $dom(V_2) = \{1,2\}$
- $\mathcal{C} = \{C_1,C_2,C_3\}$
  - C_1: $V_2 \neq 2$
  - C_2: $V_1 + V_2 < 5$
  - C_3: $V_1 > V_2$

Possible worlds for this CSP:

$\{V_1=1, V_2=1\}$
$\{V_1=1, V_2=2\}$
$\{V_1=2, V_2=1\}$ (the only model)
$\{V_1=2, V_2=2\}$
$\{V_1=3, V_2=1\}$
$\{V_1=3, V_2=2\}$

Constraints

- Constraints are restrictions on the values that one or more variables can take
  - Unary constraint: restriction involving a single variable
    - E.g.: $V_2 \neq 2$
  - k-ary constraint: restriction involving k different variables
    - E.g. binary: $V_1 + V_2 < 5$
    - E.g. 3-ary: $V_1 + V_2 + V_4 < 5$
    - We will mostly deal with binary constraints
  - Constraints can be specified by
    1. listing all combinations of valid domain values for the variables participating in the constraint
       - E.g. for constraint $V_1 > V_2$ and $dom(V_1) = \{1,2,3\}$ and $dom(V_2) = \{1,2\}$:

V_1	V_2
2	1
3	1
3	2

    2. giving a function that returns true when given values for each variable which satisfy the constraint: $V_1 > V_2$

Constraints

– Constraints can be specified by

1. listing all combinations of valid domain values for the variables participating in the constraint
   – E.g. for constraint $V_1 > V_2$ and $dom(V_1) = \{1,2,3\}$ and $dom(V_2) = \{1,2\}$:

V_1	V_2
2	1
3	1
3	2

2. giving a function that returns true when given values for each variable which satisfy the constraint: $V_1 > V_2$

- A possible world satisfies a set of constraints
  – if the values for the variables involved in each constraint are consistent with that constraint
    1. They are elements of the list of valid domain values
    2. Function returns true for those values
  – Examples
    - $\{V_1=1, V_2=1\}$ (does not satisfy above constraint)
    - $\{V_1=3, V_2=1\}$ (satisfies above constraint)

Scope of a constraint

Definition:
The scope of a constraint is the set of variables that are involved in the constraint

- Examples:
  - $V_2 \neq 2$ has scope $\{V_2\}$
  - $V_1 > V_2$ has scope $\{V_1, V_2\}$
  - $V_1 + V_2 + V_4 < 5$ has scope $\{V_1, V_2, V_4\}$

- How many variables are in the scope of a k-ary constraint ?
  k variables

Finite Constraint Satisfaction Problem: Definition

Definition:
A finite constraint satisfaction problem (CSP) is a CSP with a finite set of variables and a finite domain for each variable

We will only study finite CSPs.

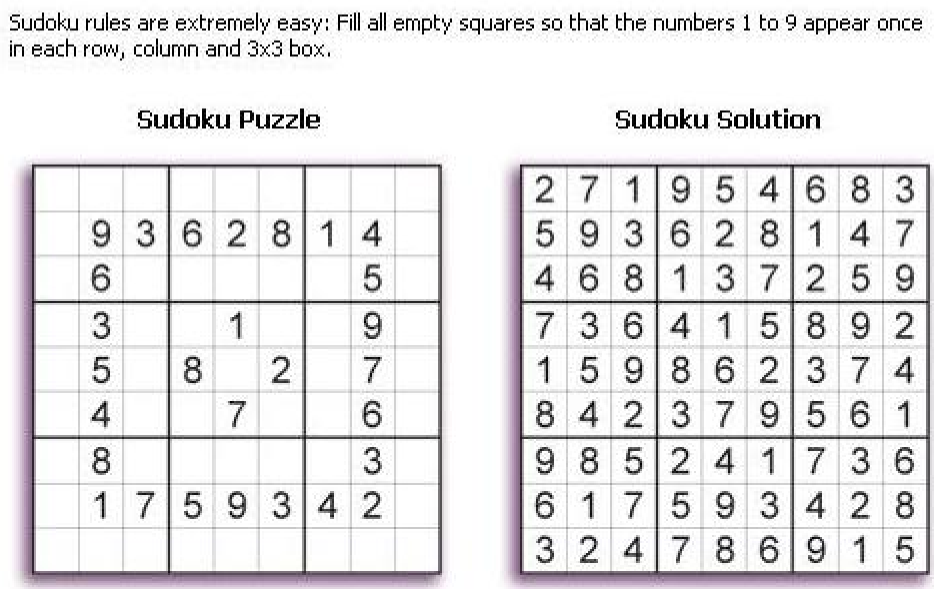
The scope of each constraint is automatically finite since it is a subset of the finite set of variables.

Examples: variables, domains, constraints

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

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

Examples: variables, domains, constraints



Sudoku Puzzle

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

Sudoku Solution

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

- Sudoku
  - variables are cells
  - domain of each variable is {1,2,3,4,5,6,7,8,9}
  - constraints: rows, columns, boxes contain all different numbers
- How many possible worlds are there? (say, 53 empty cells)

$53*9$ | 53^9 | 9^{53}

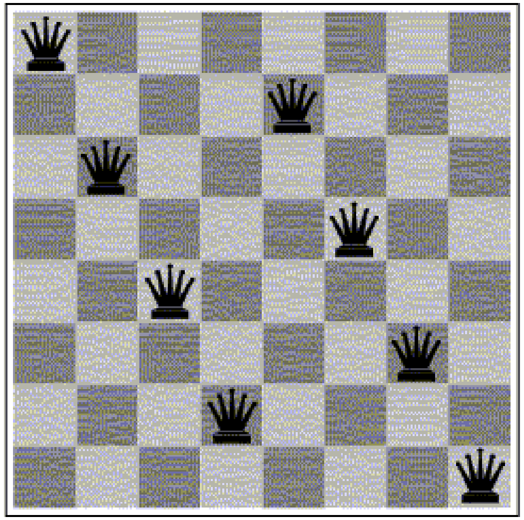
- How many models are there in a typical Sudoku?

About 2^{53} | 1 | 9^{53}

Examples: variables, domains, constraints

- 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: tasks can't be scheduled in the same location at the same time; certain tasks can't be scheduled in different locations at the same time; some tasks must come earlier than others; etc.

- 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



Constraint Satisfaction Problems: Variants

- We may want to solve the following problems with a CSP:
  - determine whether or not a model exists
  - find a model
  - find all of the models
  - count the number of models
  - find the best model, given some measure of model quality
    - this is now an optimization problem
  - determine whether some property of the variables holds in all models

Solving Constraint Satisfaction Problems

- Even the simplest problem of determining whether or not a model exists in a general CSP with finite domains is NP-hard
  - There is no known algorithm with worst case polynomial runtime
  - We can't hope to find an algorithm that is polynomial for all CSPs
- However, we can try to:
  - identify special cases for which algorithms are efficient (polynomial)
  - identify algorithms that are fast on typical cases

Lecture Overview

- Constraint Satisfaction Problems (CSPs): Definition and Recap

➡ Constraint Satisfaction Problems (CSPs): Motivation

- Solving Constraint Satisfaction Problems (CSPs)
  - Generate & Test
  - Graph search
  - Arc consistency (start)

CSP/logic: formal verification

Hardware verification (e.g., IBM)

Software verification (small to medium programs)

Most progress in the last 10 years based on:

Encodings into propositional satisfiability (SAT)

The Propositional Satisfiability Problem (SAT)

- Formula in propositional logic
  - I.e., it only contains propositional (Boolean) variables
  - Shorthand notation: x for X=true, and $\neg x$ for X=false
  - Literal: x, $\neg x$

- In so-called conjunctive normal form (CNF)
  - Conjunction of disjunctions of literals
  - E.g., $F = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

  - Let's write this as a CSP:
    - 3 variables: X_1, X_2, X_3
    - Domains: for all variables {true, false}
    - Constraints:

      $(x_1 \vee x_2 \vee x_3)$

      $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

      $(\neg x_1 \vee \neg x_2 \vee x_3)$
    - One of the models: X_1 = true, X_2 = false, X_3 = true

Importance of SAT

- Similar problems as in CSPs
  - Decide whether F has a model
  - Find a model of F
- First problem shown to be NP-hard problem
  - One of the most important problems in theoretical computer science
    - Is there an efficient (i.e. worst-case polynomial) algorithm for SAT?
      - I.e., is NP = P?
    - SAT is a deceivingly simple problem!
- Important in practice: encodings of formal verification problems
  - Software verification (finding bugs in Windows etc)
  - Hardware verification: verify computer chips (IBM big player)

SAT is one of the problems I work on

- Building algorithms that perform well in practice
  - On the type of instances we face
    - Software and hardware verification instances
    - 100000s of variables, millions of constraints
    - Runtime: seconds !
  - But: there are types of instances where current algorithms fail

- International SAT competition (http://www.satcompetition.org/)
  - About 40 solvers from around the world compete, bi-yearly
  - Best solver in 2007 and 2009:

SATzilla: a SAT solver monster

(combines many other SAT solvers)

Lin Xu, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown

(all from UBC)

Lecture Overview

- Constraint Satisfaction Problems (CSPs):
  Definition and Recap

- Constraint Satisfaction Problems (CSPs): Motivation

➡ Solving Constraint Satisfaction Problems (CSPs)
  - Generate & Test
  - Graph search
  - Arc consistency (start)

Generate and Test (GT) Algorithms

- Systematically check all possible worlds
  - Possible worlds: cross product of domains
    $dom(V_1) \times dom(V_2) \times \cdots \times dom(V_n)$

- Generate and Test:
  - Generate possible worlds one at a time
  - Test constraints for each one.

Example: 3 variables A,B,C

```
For a in dom(A)
    For b in dom(B)
        For c in dom(C)
            if {A=a, B=b, C=c} satisfies all constraints
                return {A=a, B=b, C=c}
fail
```

Generate and Test (GT) Algorithms

- If there are k variables, each with domain size d, and there are c constraints, the complexity of Generate & Test is

  $O(ckd)$ $O(ck^d)$ $O(cd^k)$ $O(d^{ck})$

  - There are d^k possible worlds
  - For each one need to check c constraints

Lecture Overview

- Constraint Satisfaction Problems (CSPs): Definition and Recap
- Constraint Satisfaction Problems (CSPs): Motivation
- Solving Constraint Satisfaction Problems (CSPs)
  - Generate & Test
  - Graph search
  - Arc consistency (start)

CSP as a Search Problem: one formulation

- States: partial assignment of values to variables
- Start state: empty assignment
- Successor function: states with the next variable assigned
  - E.g., follow a total order of the variables $V_1, \ldots, V_n$
  - A state assigns values to the first k variables:
    - $\{V_1 = v_1, \ldots, V_k = v_1\}$
    - Neighbors of node $\{V_1 = v_1, \ldots, V_k = v_1\}$:
      nodes $\{V_1 = v_1, \ldots, V_k = v_{1,} V_{k+1} = x\}$ for each $x \in dom(V_{k+1})$

- Goal state: complete assignments of values to variables that satisfy all constraints
  - That is, models
- Solution: assignment (the path doesn't matter)

Which search algorithm would be most appropriate for this formulation of CSP?

Depth First Search

Least Cost First Search

A *

None of the above

Relationship To Search

- The path to a goal isn't important, only the solution is

- Heuristic function: "none"
  - All goals are at the same depth

- CSP problems can be huge
  - Thousands of variables
    - Exponentially more search states
  - Exhaustive search is typically infeasible

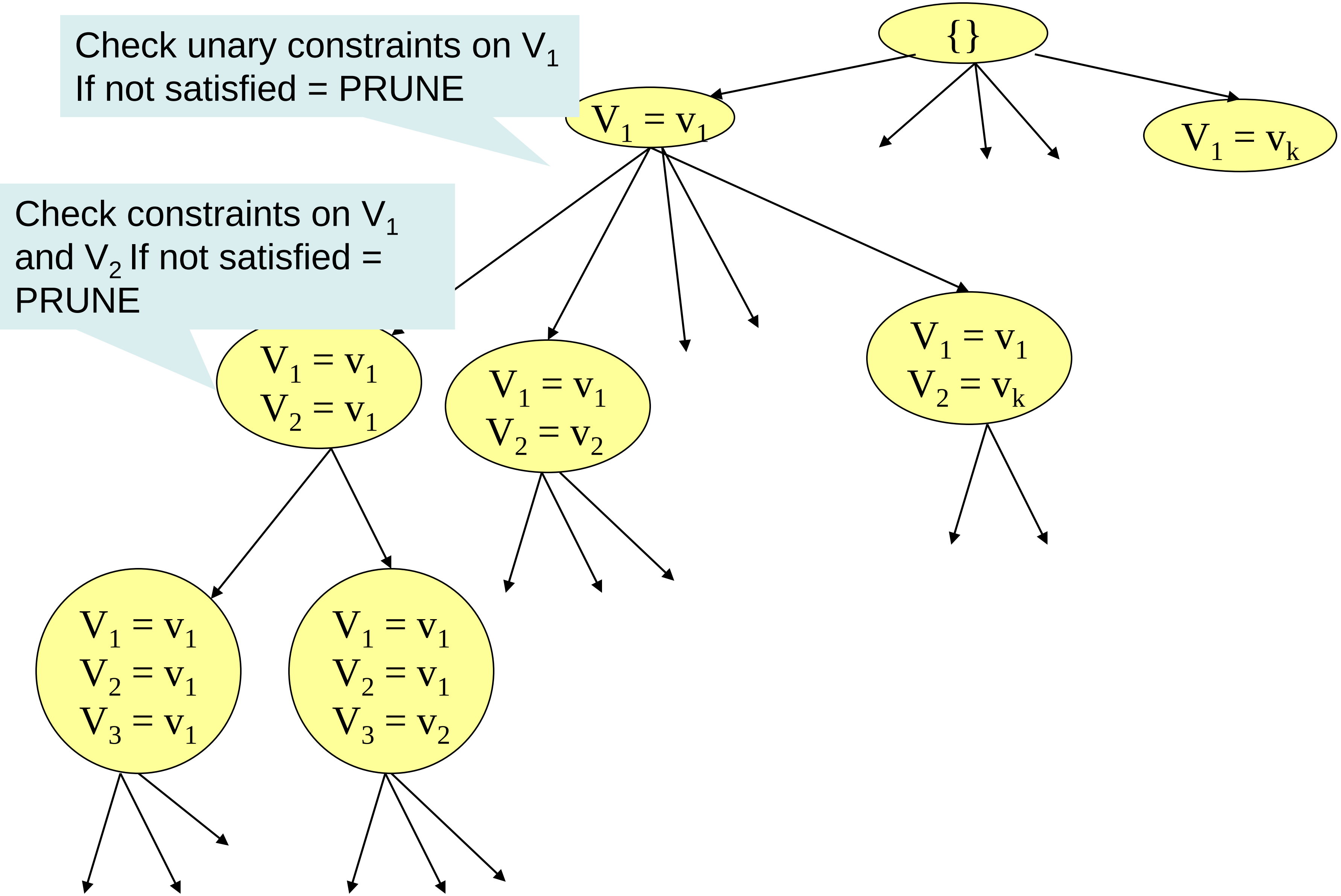
- Many algorithms exploit the structure provided by the goal $\Rightarrow$ set of constraints, *not* black box

Backtracking algorithms

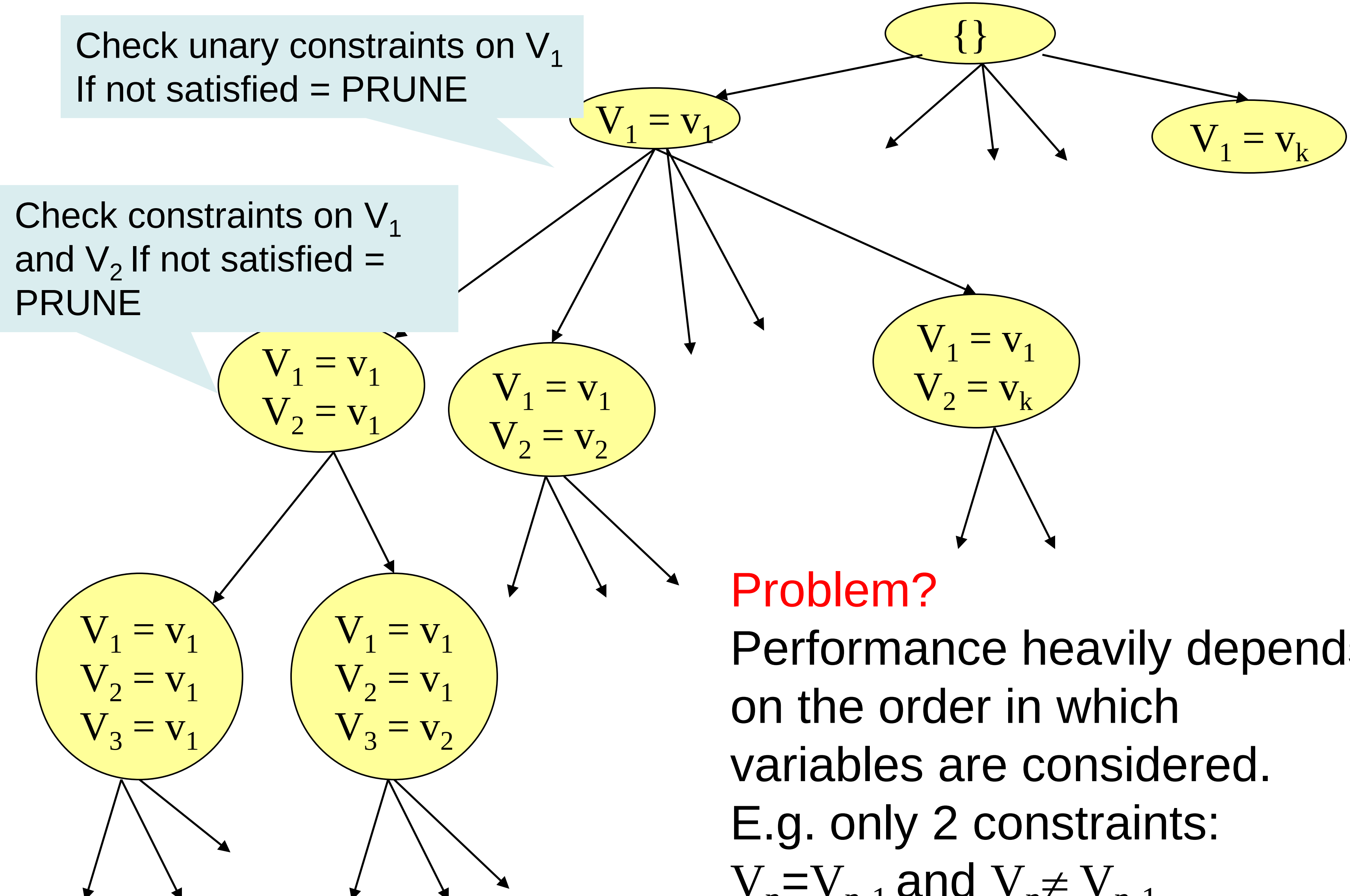
- Explore search space via DFS but evaluate each constraint as soon as all its variables are bound.
- Any partial assignment that doesn't satisfy the constraint can be pruned.
- Example:
  - 3 variables A, B,C, each with domain {1,2,3,4}
  - {A = 1, B = 1} is inconsistent with constraint $A \neq B$ regardless of the value of the other variables

    $\Rightarrow$ Prune!

CSP as Graph Searching



CSP as Graph Searching



Problem?

Performance heavily depends on the order in which variables are considered.

E.g. only 2 constraints:

$V_n = V_{n-1}$ and $V_n \neq V_{n-1}$

CSP as a Search Problem: another formulation

- States: partial assignment of values to variables
- Start state: empty assignment
- Successor function: states with the next variable assigned
  - Assign any previously unassigned variable
  - A state assigns values to some subset of variables:
    - E.g. $\{V_7 = v_1, V_2 = v_1, V_{15} = v_1\}$
    - Neighbors of node $\{V_7 = v_1, V_2 = v_1, V_{15} = v_1\}$:
      nodes $\{V_7 = v_1, V_2 = v_1, V_{15} = v_1, V_x = y\}$
      for any variable $V_x \in \mathcal{V} \setminus \{V_7, V_2, V_{15}\}$ and any value $y \in dom(V_x)$
- Goal state: complete assignments of values to variables that satisfy all constraints
  - That is, models
- Solution: assignment (the path doesn't matter)

Selecting variables in a smart way

- Backtracking relies on one or more heuristics to select which variables to consider next
  - E.g, variable involved in the highest number of constraints
  - Can also be smart about which values to consider first
- This is a different use of the word “heuristic” !
  - Still true in this context
    - Can be computed cheaply during the search
    - Provides guidance to the search algorithm
  - But not true anymore in this context
    - “Estimate of the distance to the goal”
- Both meanings are used frequently in the AI literature

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: all variables assigned a value and all constraints satisfied?
  - 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

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)
- Implement the Generate-and-Test Algorithm. Explain its disadvantages.
- Solve a CSP by search (specify neighbors, states, start state, goal state). Compare strategies for CSP search. Implement pruning for DFS search in a CSP.

- Coming up: Arc consistency and domain splitting
  - Read Sections 4.5-4.6
- Assignment 1 is due next Monday