Arc Consistency

CPSC 322 – CSP 3

Textbook § 4.5

February 2, 2011

Lecture Overview

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

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	***Decision Networks*** Variable Elimination ***Markov Processes*** Value Iteration	Decision Theory

We'll now focus on CSP

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

Definition:
A model of a CSP is a possible world that satisfies all constraints.

An example CSP:

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

Possible worlds for this CSP:

$\{V_1=1, V_2=1\}$
$\{V_1=1, V_2=2\}$
$\{V_1=2, V_2=1\}$ (one model)
$\{V_1=2, V_2=2\}$
$\{V_1=3, V_2=1\}$ (another model)
$\{V_1=3, V_2=2\}$

Generate and Test (GT) Algorithms

- 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
```

- Simple, but slow:
  - k variables, each domain size d, c constraints: $O(cd^k)$

Lecture Overview

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

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.

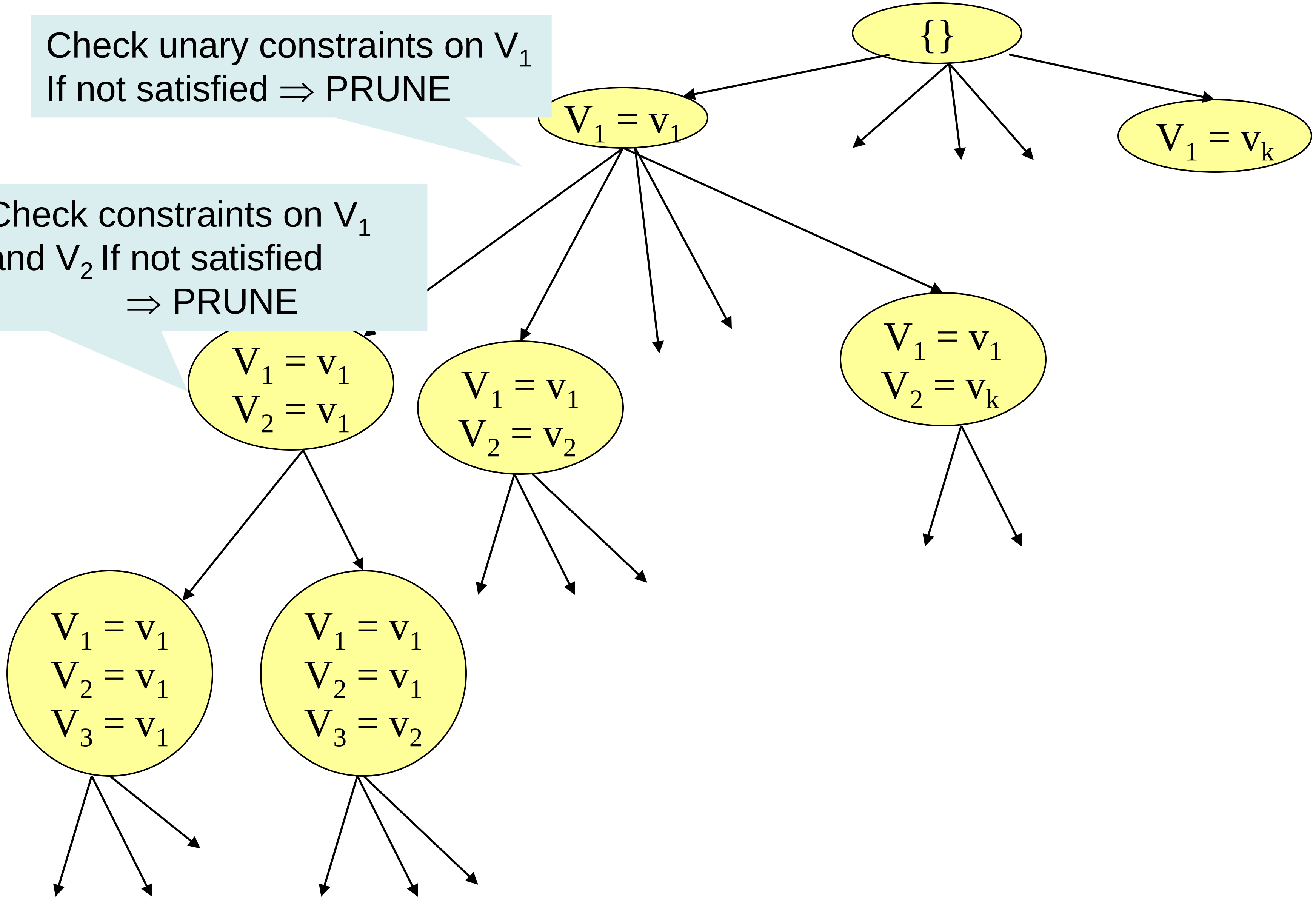
- Example:
  - 3 variables A, B,C, each with domain {1,2,3,4}
  - {A = 1, B = 1} is inconsistent with constraint $A \neq B$ regardless of the value of the other variables

    $\Rightarrow$ Prune!

CSP as Graph Searching

Check unary constraints on V_1
If not satisfied $\Rightarrow$ PRUNE

Check constraints on V_1 and V_2 If not satisfied $\Rightarrow$ PRUNE



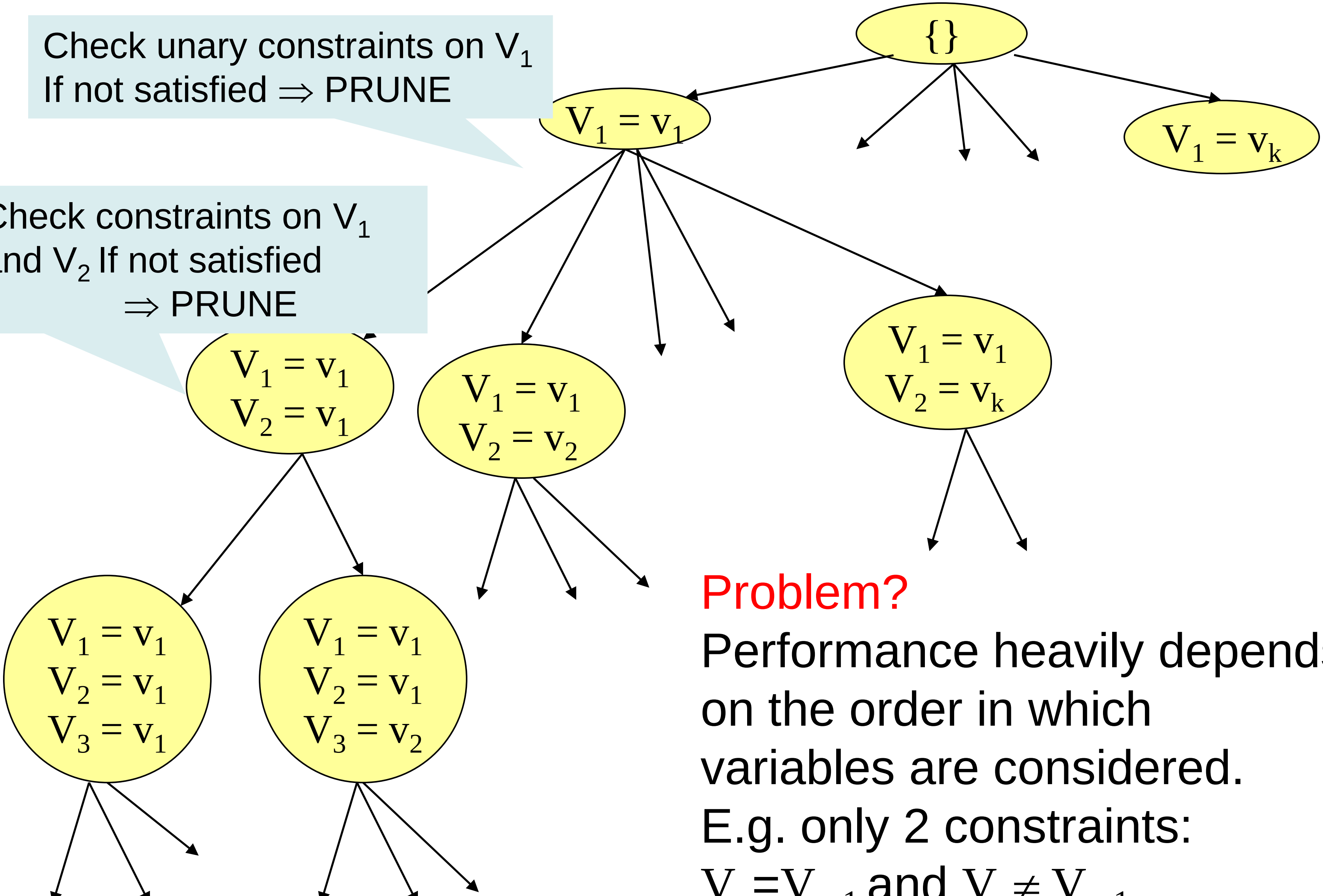
Standard Search vs. Specific R&R systems

- 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

CSP as Graph Searching

Check unary constraints on V_1
If not satisfied $\Rightarrow$ PRUNE

Check constraints on V_1 and V_2 If not satisfied $\Rightarrow$ PRUNE



Problem?

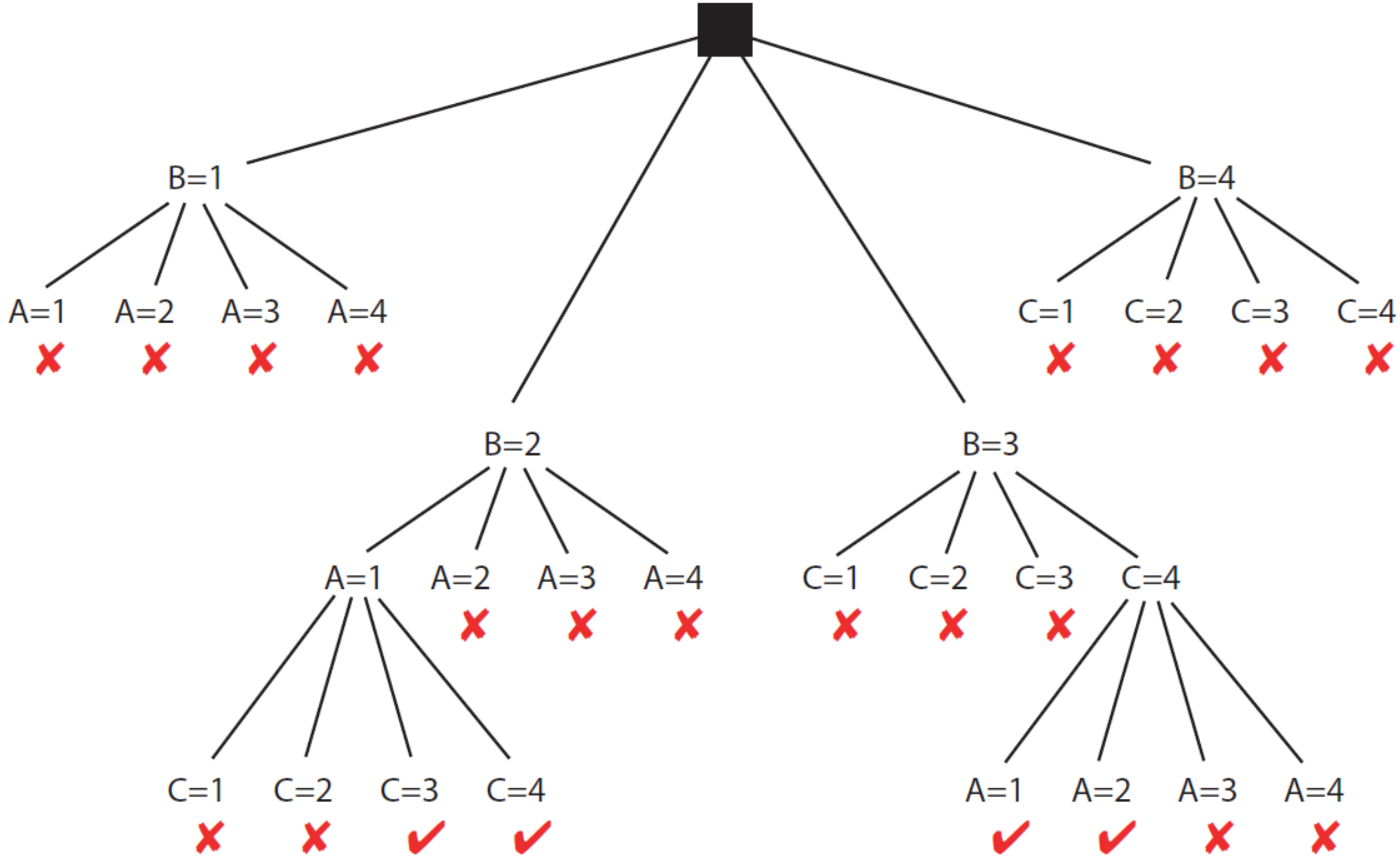
Performance heavily depends on the order in which variables are considered.
E.g. only 2 constraints:
$V_n = V_{n-1}$ and $V_n \neq V_{n-1}$

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)

CSP as Graph Searching

- 3 Variables: A,B,C. All with domains = {1,2,3,4}
- Constraints: A<B, B<C



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

Learning Goals for solving CSPs so far

- 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.

Lecture Overview

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

Can we do better than Search?

Key idea

- prune the domains as much as possible before "searching" for a solution.

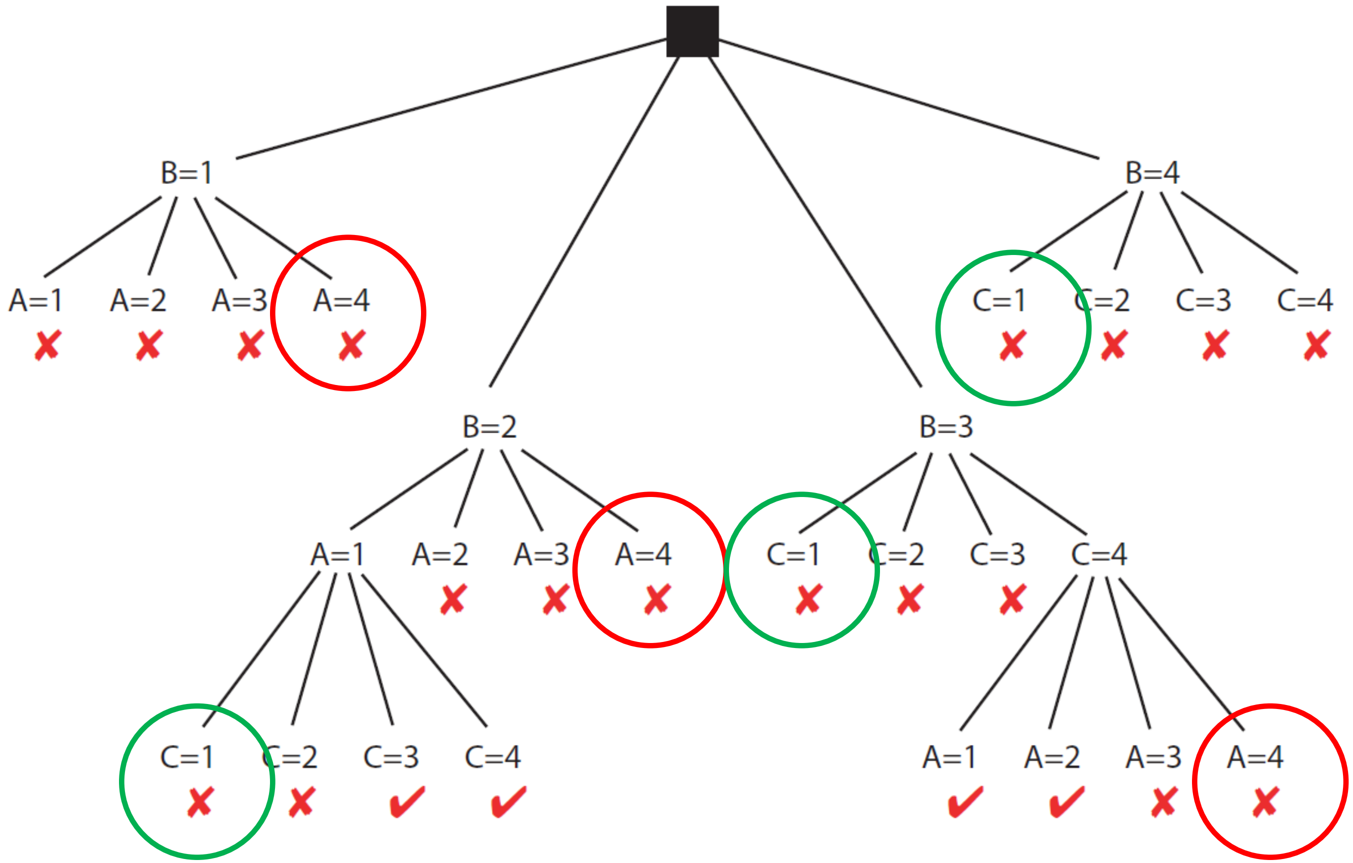
Def.: A variable is domain consistent if no value of its domain is ruled impossible by any unary constraints.

- Example: $dom(V_2) = \{1, 2, 3, 4\}$. $V_2 \neq 2$
- Variable V_2 is not domain consistent.
  - It is domain consistent once we remove 2 from its domain.

- Trivial for unary constraints. Trickier for k-ary ones.

Graph Searching Redoes Work

- 3 Variables: A,B,C. All with domains = {1,2,3,4}
- Constraints: A<B, B<C
- A ≠ 4 is rediscovered 3 times. So is C ≠ 1
  - Solution: remove values from A’s and C’s domain once and for all

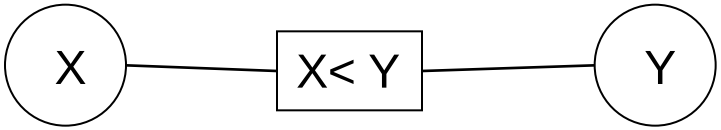


Constraint network: definition

Def. A constraint network is defined by a graph, with

- one node for every variable (drawn as circle)
- one node for every constraint (drawn as rectangle)
- undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

- Example:
  - Two variables X and Y
  - One constraint: X<Y



Constraint network: definition

Def. A constraint network is defined by a graph, with

- one node for every variable (drawn as circle)
- one node for every constraint (drawn as rectangle)
- undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

- Whiteboard example:
  - 3 Variables A,B,C
  - 3 Constraints: A<B, B<C, A+3=C
  - 6 edges in the constraint network:
    - ⟨A,A<B⟩ , ⟨B,A<B⟩
    - ⟨B,B<C⟩ , ⟨C,B<C⟩
    - ⟨A, A+3=C⟩ , ⟨C,A+3=C⟩

A more complicated example

- How many variables are there in this constraint network?

5

6

9

14

  - Variables are drawn as circles

- How many constraints are there?

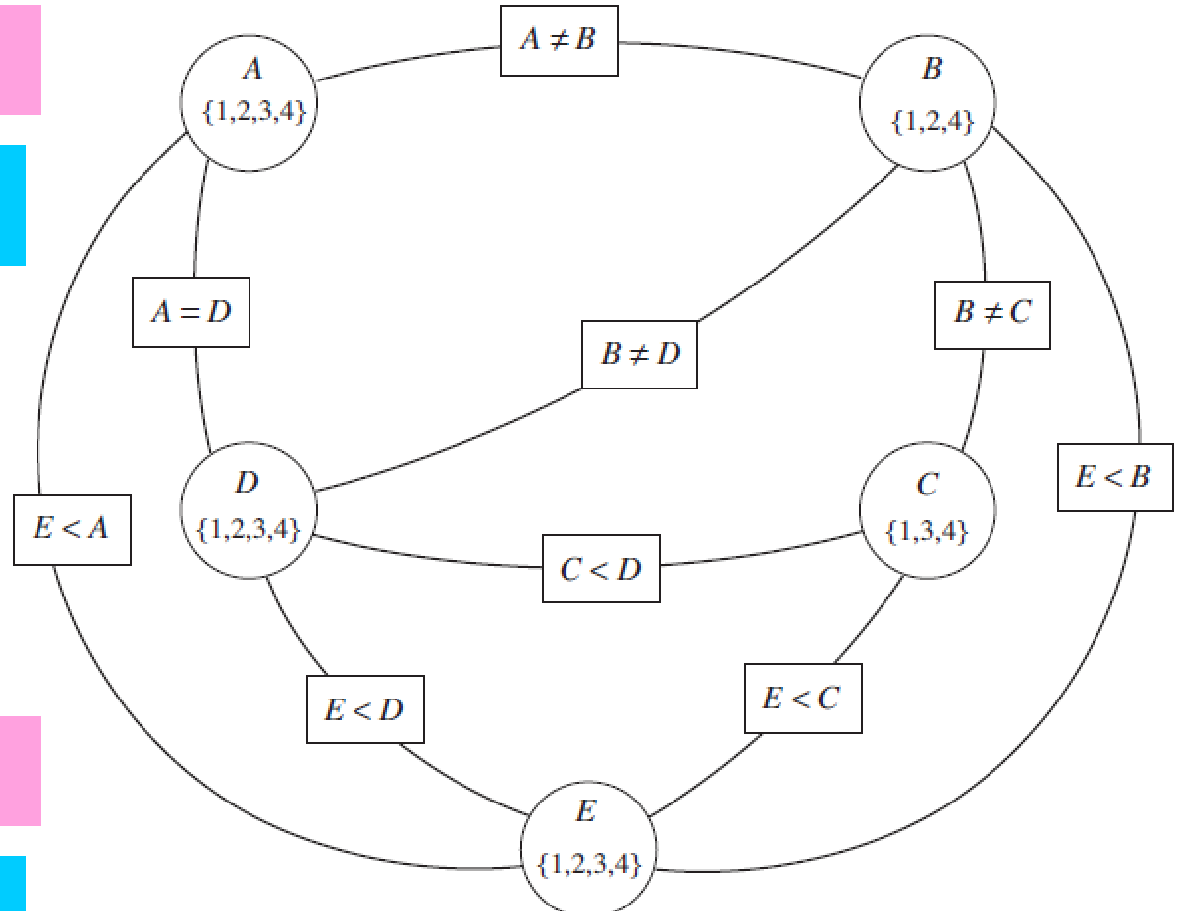
5

6

9

14

  - Constraints are drawn as rectangles

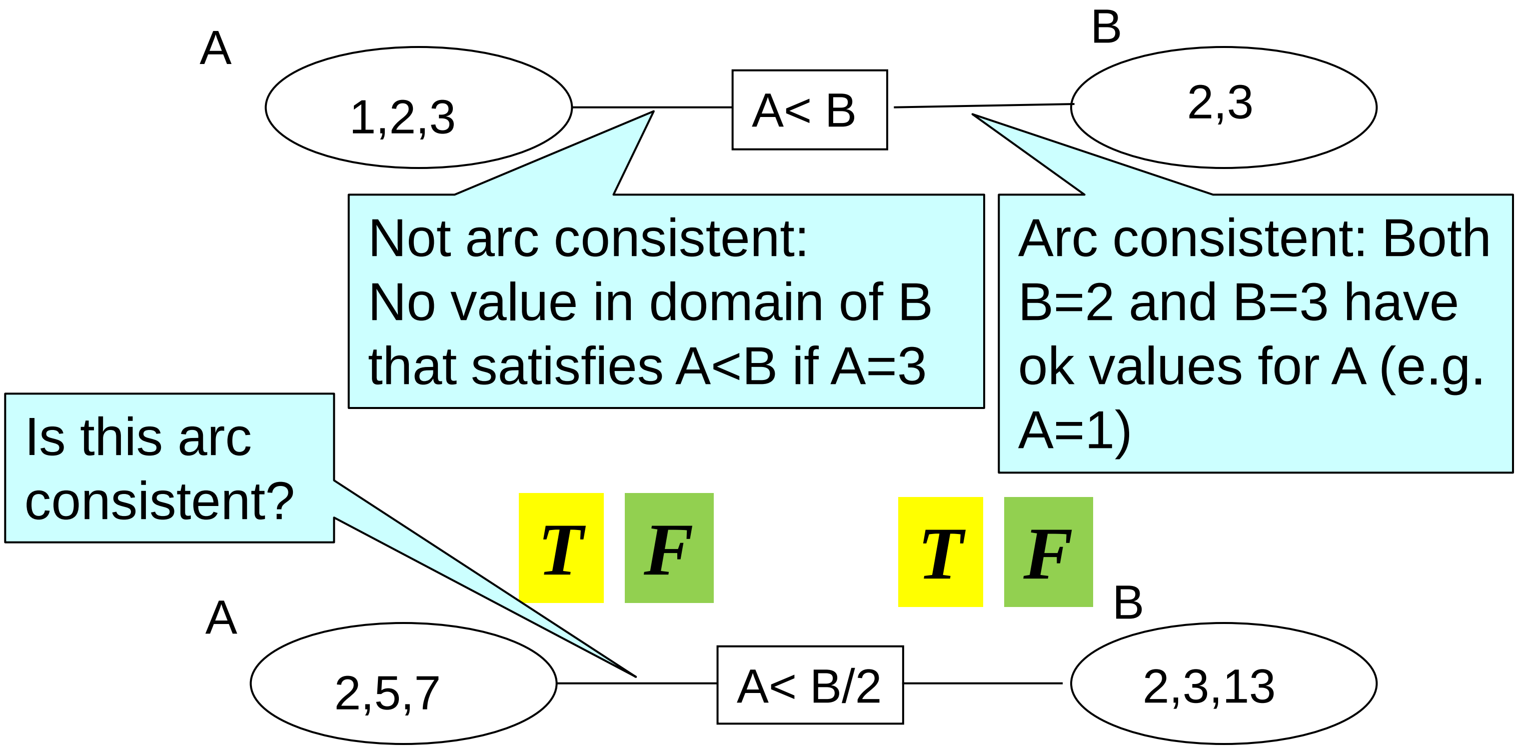


Arc Consistency

Definition:

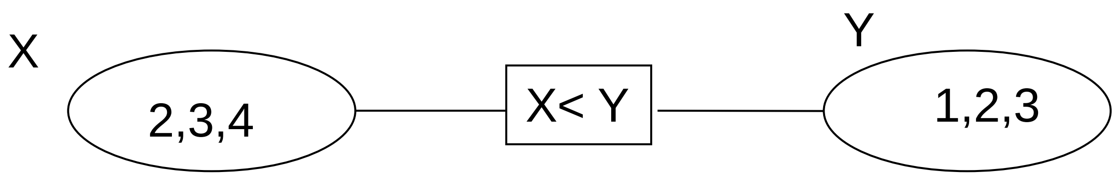
An arc <x, r(x,y)> is arc consistent if for each value x in dom(X) there is some value y in dom(Y) such that r(x,y) is satisfied.

A network is arc consistent if all its arcs are arc consistent.



How can we enforce Arc Consistency?

- If an arc <*X, r(X,Y)*> is not arc consistent
  - Delete all values *x* in *dom(X)* for which there is no corresponding value in *dom(Y)*
  - This deletion makes the arc <X, r(X,Y)> arc consistent.
  - This removal can never rule out any models/solutions
    - Why?



http://cs.ubc.ca/~hutter/teaching/cpsc322/aispace/simple-network.xml

Arc Consistency Algorithm: high level strategy

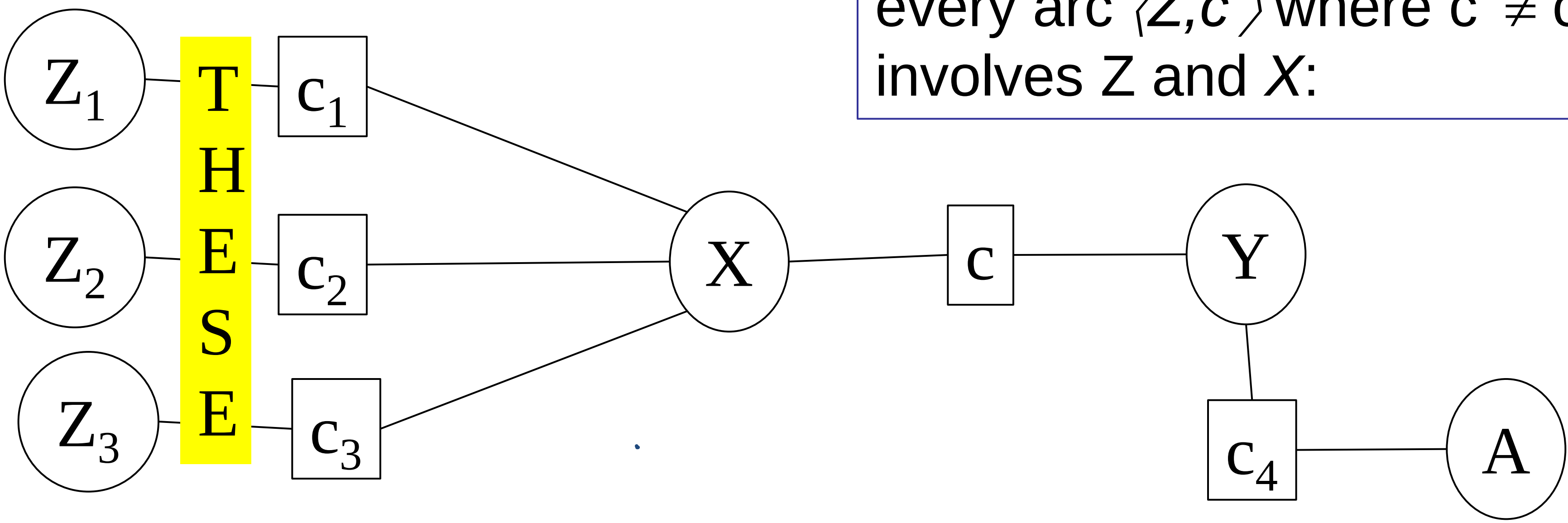
- Consider the arcs in turn, making each arc consistent
- Reconsider arcs that could be made inconsistent again by this pruning

- See “simple problem 1” in AIspace for an example:

Which arcs need to reconsidered?

- When we reduce the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, which arcs do we need to reconsider?

every arc ⟨*Z,c'*⟩ where c' ≠ c involves Z and *X*:



- You do not need to reconsider other arcs
  - If an arc ⟨X,c'⟩ was arc consistent before, it will still be arc consistent
  - Nothing changes for arcs of constraints not involving X

Which arcs need to reconsidered?

- Consider the arcs in turn, making each arc consistent
- Reconsider arcs that could be made inconsistent again by this pruning

- Trace on “simple problem 1” and on
  “scheduling problem 1”, trying to predict

- which arcs are not consistent and
- which arcs need to be reconsidered after each removal

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

$\mathbf{D}_X$ is a set of values for each variable X

TDA is a set of arcs

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

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

return $\{\mathbf{D}_X \mid X \text{ is a variable}\}$

TDA: ToDoArcs, blue arcs in AIspace

Scope of constraint c is the set of variables involved in that constraint

ND_X: values x for X for which there a value for y supporting x

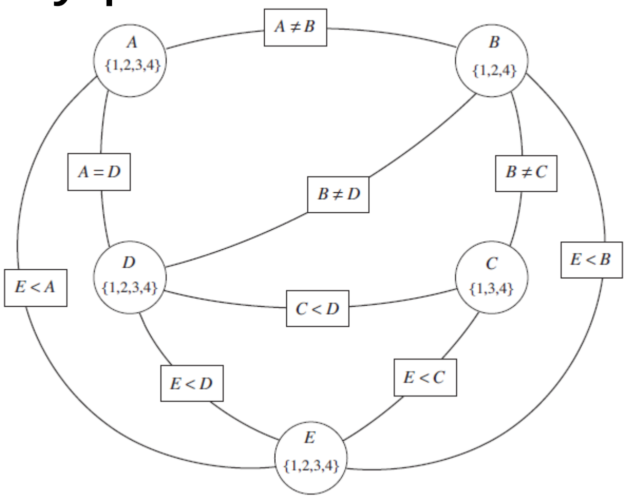
X's domain changed: $\Rightarrow$ arcs (Z,c') for variables Z sharing a constraint c' with X could become inconsistent

Arc Consistency Algorithm: Interpreting Outcomes

- Three possible outcomes
  (when all arcs are arc consistent):
  - Each domain has a single value, e.g.
    http://cs.ubc.ca/~hutter/teaching/cpsc322/aispace/simple-network.xml
    - We have a (unique) solution.
  - At least one domain is empty, e.g.
    http://cs.ubc.ca/~hutter/teaching/cpsc322/aispace/simple-infeasible.xml
    - No solution! All values are ruled out for this variable.
  - Some domains have more than one value, e.g.
    built-in example “simple problem 2”
    - There may be a solution, multiple ones, or no one
    - Need to solve this new CSP problem:
      same constraints, domains have been reduced

Arc Consistency Algorithm: Complexity

- Worst-case complexity of arc consistency procedure on a problem with N variables
  – let ***d*** be the max size of a variable domain
  – let ***c*** be the number of constraints



  - How often will we prune the domain of variable V? O(d) times
  - How many arcs will be put on the ToDoArc list when pruning domain of variable V?
    - O(degree of variable V)
    - In total, across all variables: sum of degrees of all variables = …
      - 2*number of constraints, i.e. 2*c
  - Together: we will only put O(dc) arcs on the ToDoArc list
  - Checking consistency is $O(d^2)$ for each of them
- Overall complexity: $O(cd^3)$
- Compare to $O(d^N)$ of DFS!! Arc consistency is MUCH faster

Learning Goals for arc consistency

- Define/read/write/trace/debug the arc consistency algorithm.
- Compute its complexity and assess its possible outcomes

- Arc consistency practice exercise is on WebCT
- Coming up: Domain splitting
  - I.e., combining arc consistency and search
  - Read Section 4.6
- Also coming up: local search, Section 4.8

- Assignment 1 is due next Monday