CSPs: Search and Arc Consistency

Computer Science cpsc322, Lecture 12

(Textbook Chpt 4.3-4.5)

January, 29, 2010

Lecture Overview

- **Recap CSPs**
- Generate-and-Test
- Search
- Consistency
- Arc Consistency

Constraint Satisfaction Problems: definitions

Definition (Constraint Satisfaction Problem)

A constraint satisfaction problem consists of # possible worlds $3 * 5 * 5$

- a set of variables A, B, C
- a domain for each variable $dom\, C = \{1, 2, 3\}$ $dom\, A = \{1, 2, 3, 4, 5\} = dom(B)$
- a set of constraints $B = 5$ $C > B$ $A = B$ no sol

$''$ $C < B$ $''$

Definition (model / solution)

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

$B = 5$ $A = 5$ $C = 1$

$''$ $''$ $C = 2$

$''$ $''$ $C = 3$

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

Environment

Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency Search → for CSP	
Static	Query	*Logics* Search → for Inference	*Belief Nets* Var. Elimination
Sequential	Planning	*STRIPS* actions Search → for complex planning	*Decision Nets* Var. Elimination *Markov Processes* Value Iteration

Representation

Reasoning Technique

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

- State & start state
- Successor function
- Goal test
- Solution
- Heuristic function

Planning :

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Answering Queries

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Lecture Overview

- Recap **CSPs**
- Generate-and-Test
- Search
- Consistency
- Arc Consistency

Generate-and-Test Algorithm

- **Algorithm:**
  - Generate possible worlds one at a time
  - Test them to see if they violate any constraints

domA = {1,2,3,4,5}
domB = {1,2,3,4,5}
domC = {1,2,3}

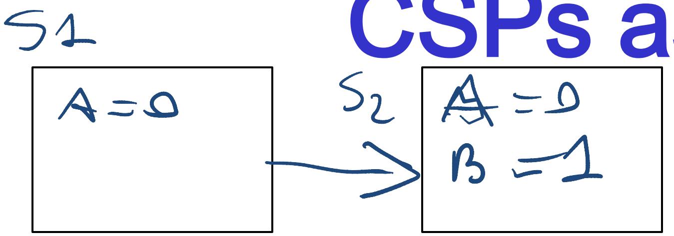
```
For a in domA
   For b in domB
       For c in domC
       if (a b c) satisfies all constraints
       return (abc)
return fail
```

- This procedure is able to solve any CSP
- However, the running time is proportional to the number of possible worlds
  - always exponential in the number of variables
  - far too long for many CSPs ☹

Lecture Overview

- Recap
- Generate-and-Test
- Search
- Consistency
- Arc Consistency

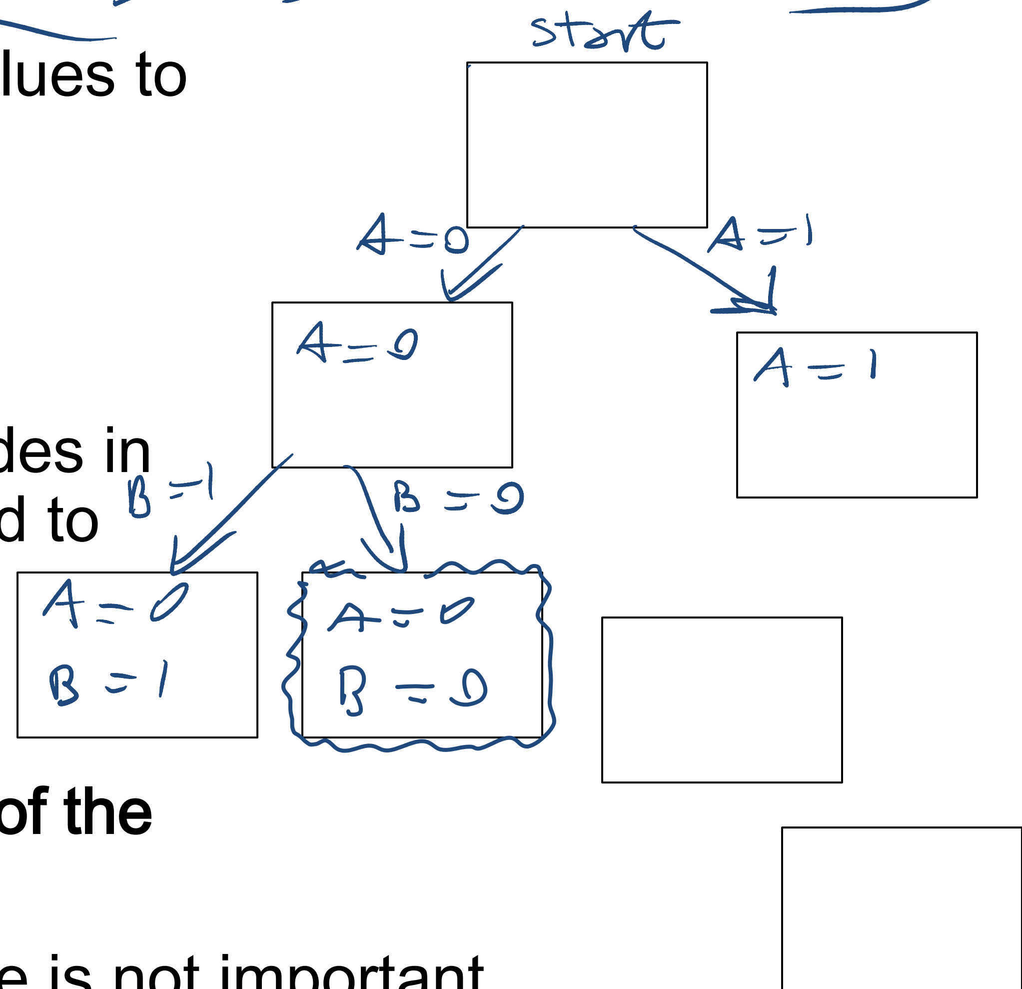
CSPs as search problems



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

$A = B$

- **states:** assignments of values to a subset of the variables
- **start state:** the empty assignment (no variables assigned values)
- **neighbours** of a state: nodes in which values are assigned to one additional variable
- **goal state:** a state which **assigns a value to each variable**, and **satisfies all of the constraints**



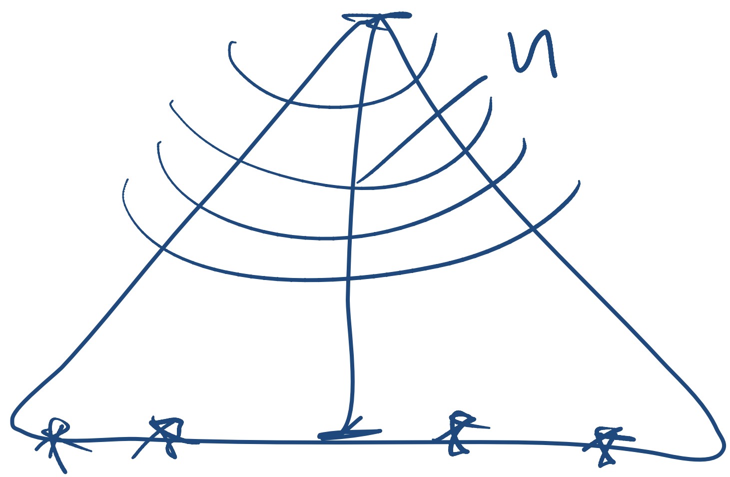
Note: the path to a goal node is not important

CSPs as Search Problems

What search strategy will work well for a CSP?

- If there are n variables every solution is at depth…n…
- Is there a role for a heuristic function?

- the tree is always …finite… and has no …cycles…, so which one is better ~~BFS~~ or ~~IDS~~ or DFS?

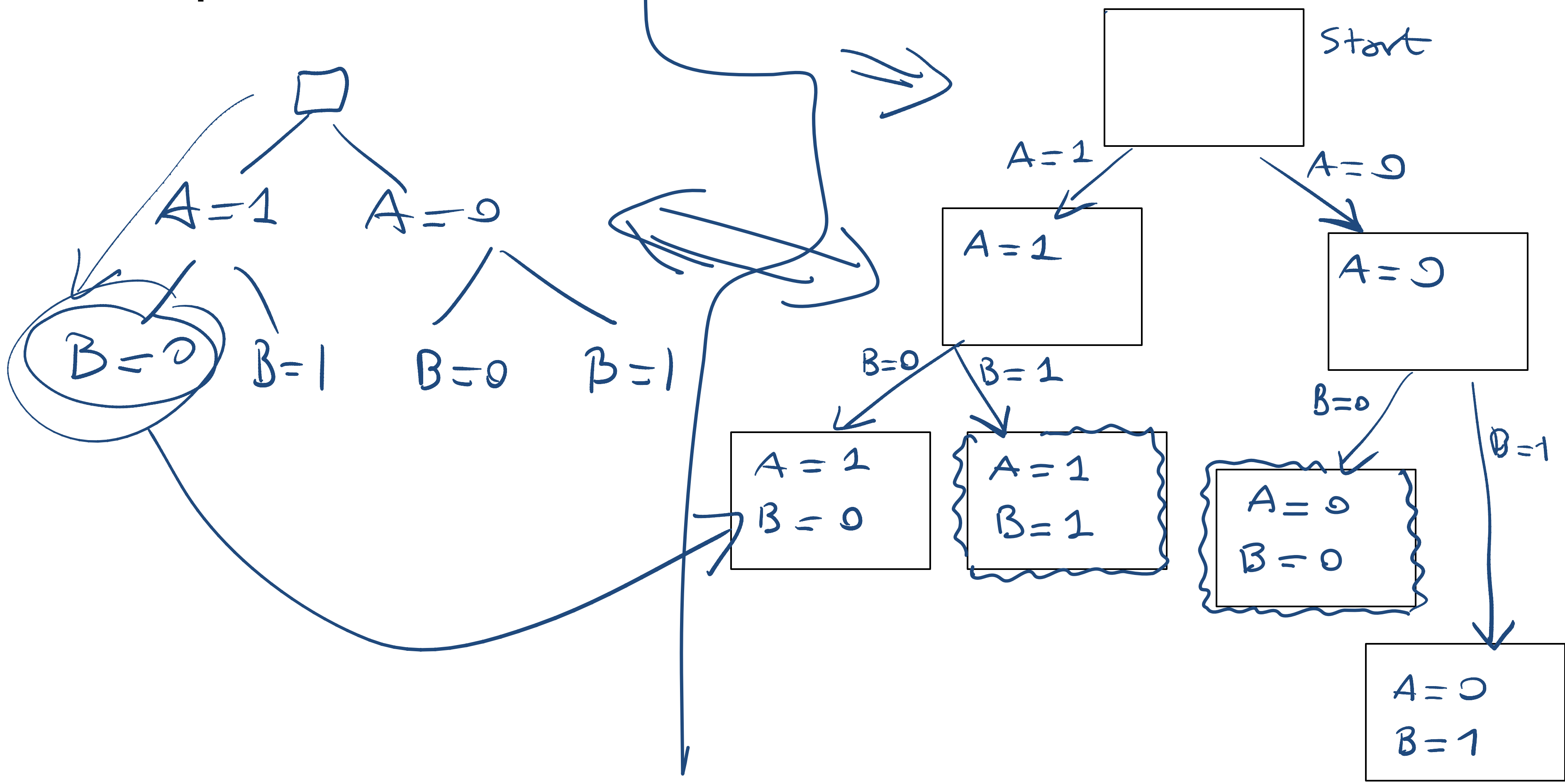


CSPs as search problems

$A, B \quad dom\ A = dom\ B = \{0, 1\}$

$const\ A = B$

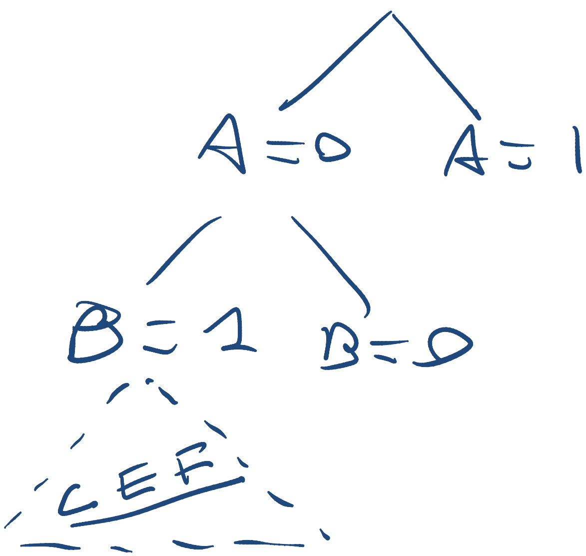
Simplified notation



CSPs as Search Problems

How can we avoid exploring some sub-trees i.e., prune the DFS Search tree?

- once we consider a path whose end node violates one or more constraints, we know that a solution cannot exist below that point
- thus we should **remove that path** rather than continuing to search



A B C E F {1, 0}

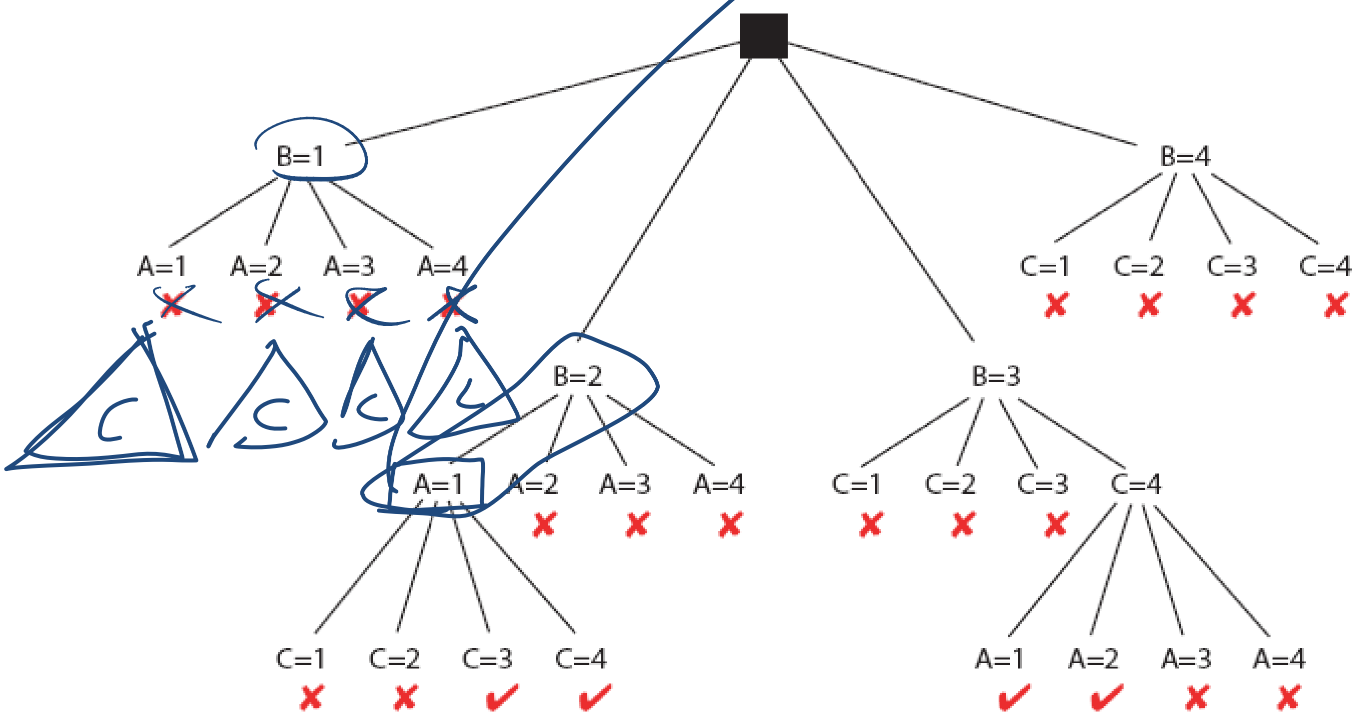
constraints (A = B) (F > A) (C = E)

Solving CSPs by DFS: Example

Problem:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C

B = 2
A = 1



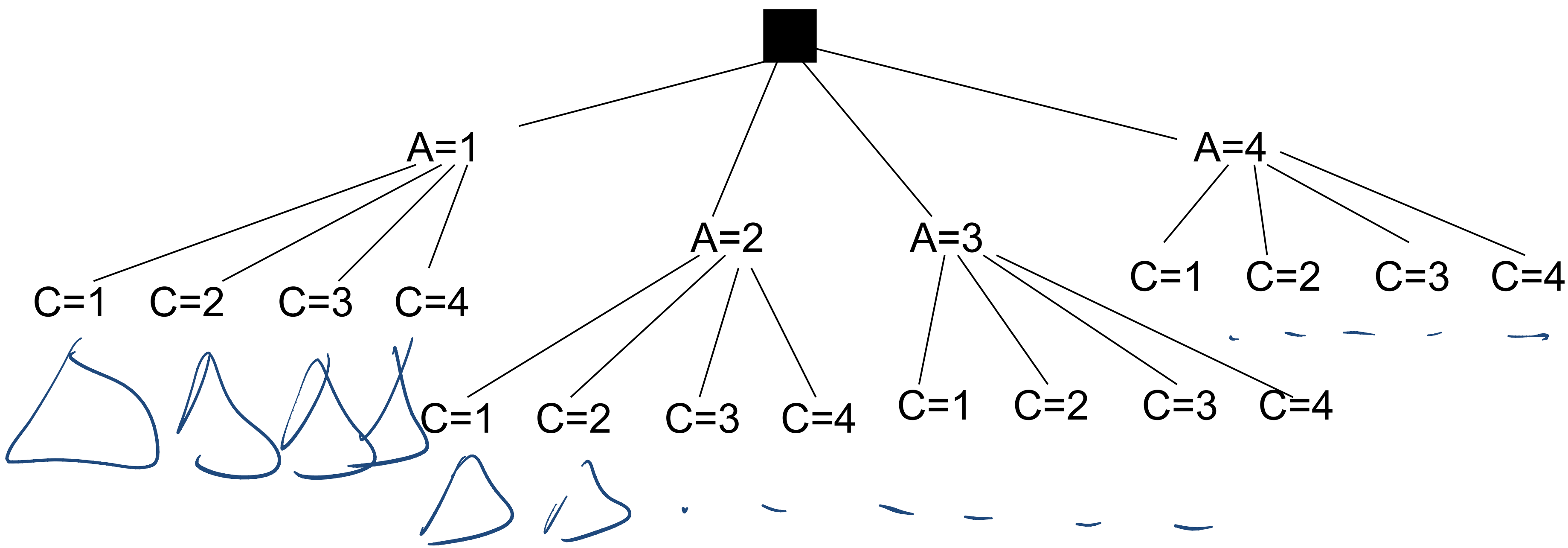
Solving CSPs by DFS: Example Efficiency

Problem:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C

Note: the algorithm's efficiency depends on the order in which variables are expanded

Degree "Heuristics"



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
- Successor function
- Goal test
- Solution
- Heuristic function

Inference

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Lecture Overview

- Recap
- Generate-and-Test Recap
- Search
- Consistency
- Arc Consistency

Can we do better than Search?

Key ideas:

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

Simple when using constraints involving single variables (technically enforcing **domain consistency**)

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

- Example: $D_B = \{1, 2, \cancel{3}, 4\}$ is not domain consistent if we have the constraint $B \neq 3$.

How do we deal with constraints involving multiple variables?

Definition (**constraint network**)

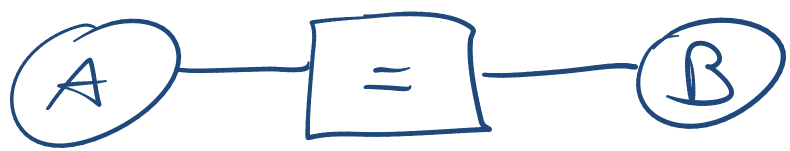
A constraint network is defined by a graph, with

- one **node** for every **variable**
- one **node** for every **constraint**

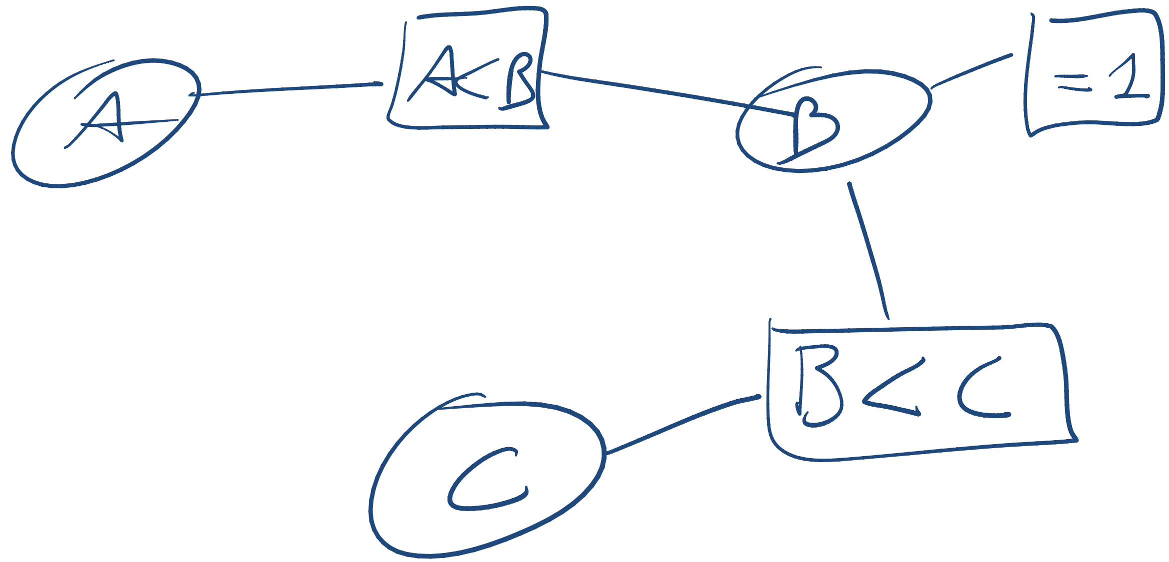
and undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

A B $\{0,1\}$

$A = B$



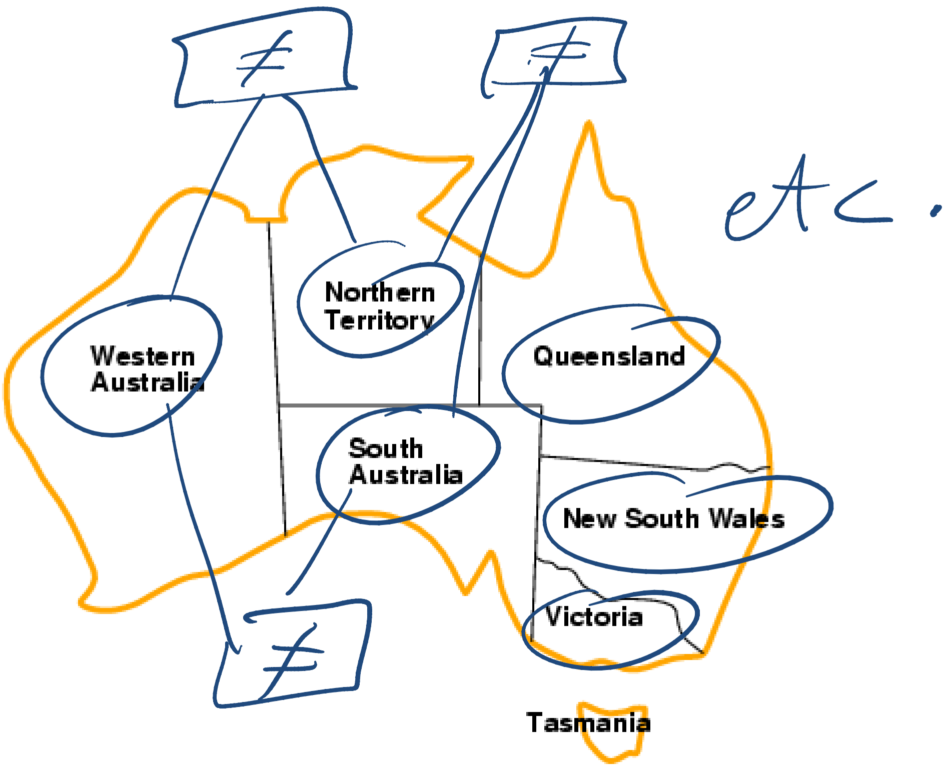
Example Constraint Network



Recall Example:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C , B = 1

Example: Constraint Network for Map-Coloring



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

Domains D_i = {red,green,blue}

Constraints: adjacent regions must have different colors

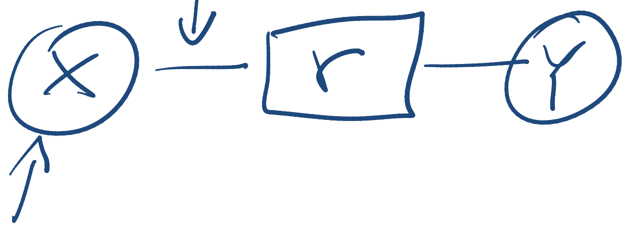
Lecture Overview

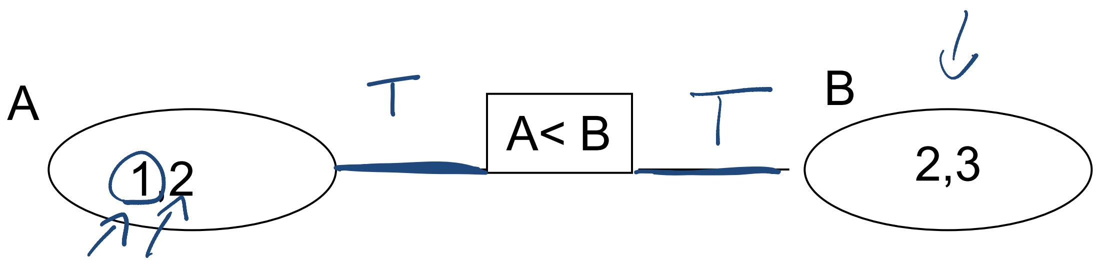
- Recap
- Generate-and-Test Recap
- Search
- Consistency
- Arc Consistency

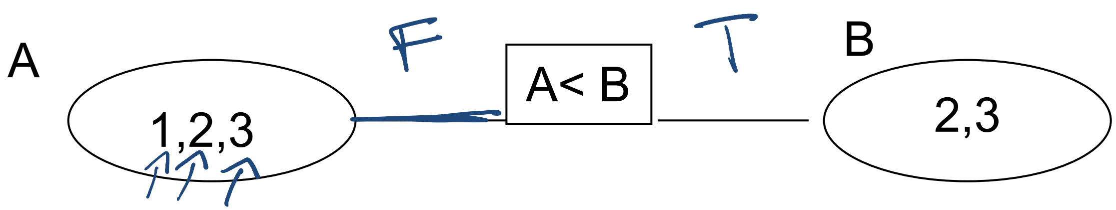
Arc Consistency

Definition (arc consistency)

An arc $\langle X, r(X,Y) \rangle$ 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.

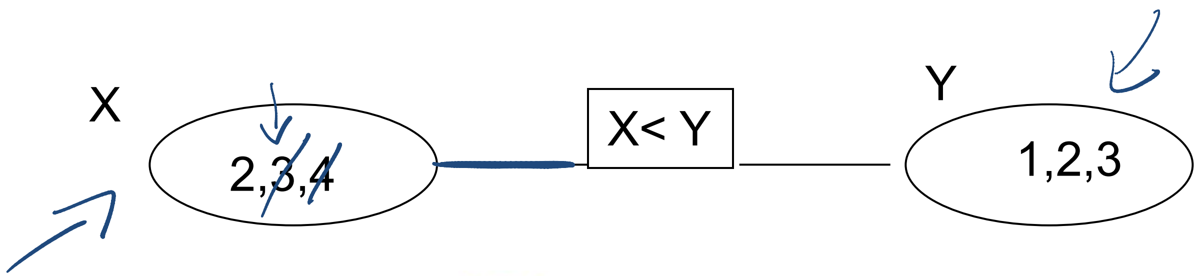






How can we enforce Arc Consistency?

- If an arc $\langle X, r(X,Y)\rangle$ is not arc consistent, all values x in $dom(X)$ for which there is no corresponding value in $dom(Y)$ may be deleted from $dom(X)$ to make the arc $\langle X, r(X,Y)\rangle$ consistent.
  - This removal can never rule out any models/solutions



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

Learning Goals for today's class

You can:

- 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.
- Build a constraint network for a set of constraints.
- Verify whether a network is arc consistent.
- Make an arc arc-consistent.

Next class

How to make a constraint network arc consistent?

Arc Consistency Algorithm

will be posted

CSP Practice Exercise ~~posted~~: check it out!