CSPs: Search and Arc Consistency

CPSC 322 Lecture 11

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

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

Definition (model / solution)

A model of a CSP is an assignment of values to variables (i.e. **possible worlds**) that satisfies all of the constraints.

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.

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

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

- 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

- **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?

A. Yes B. No

iclicker.

The search space is always…

A. Finite with cycles B. Infinite without cycles

C. Finite without cycles D. Infinite with cycles

iclicker.

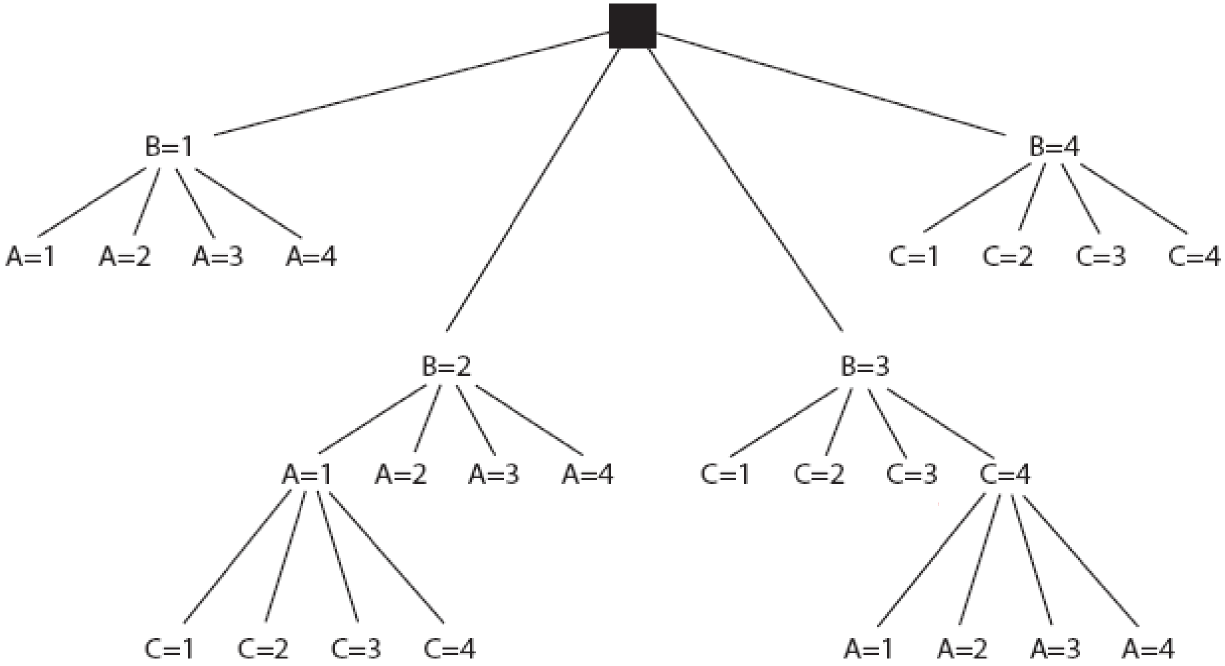
So which search strategy is better?

A. BFS
B. IDS
C. A*
D. DFS

iclicker.

CSPs as search problems

Simplified notation



CSPs as Search Problems

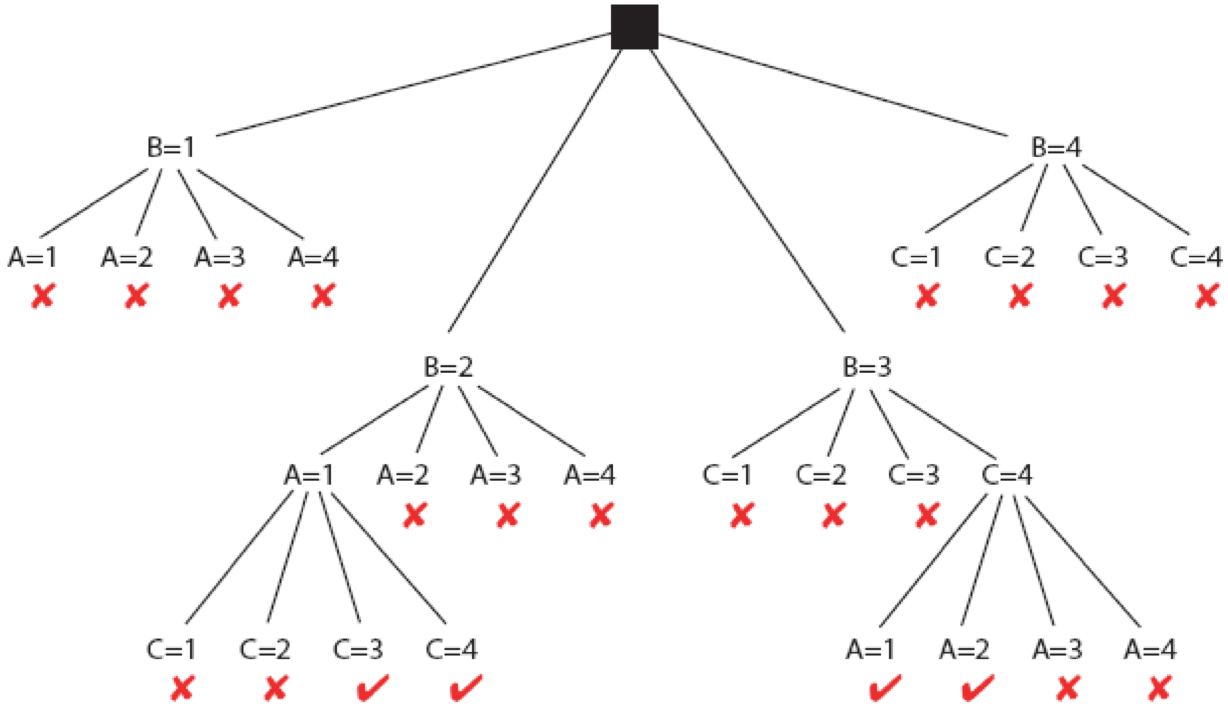
How can we avoid exploring some sub-trees i.e., prune the DFS Search tree?

- 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

Solving CSPs by DFS: Example

Problem:

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



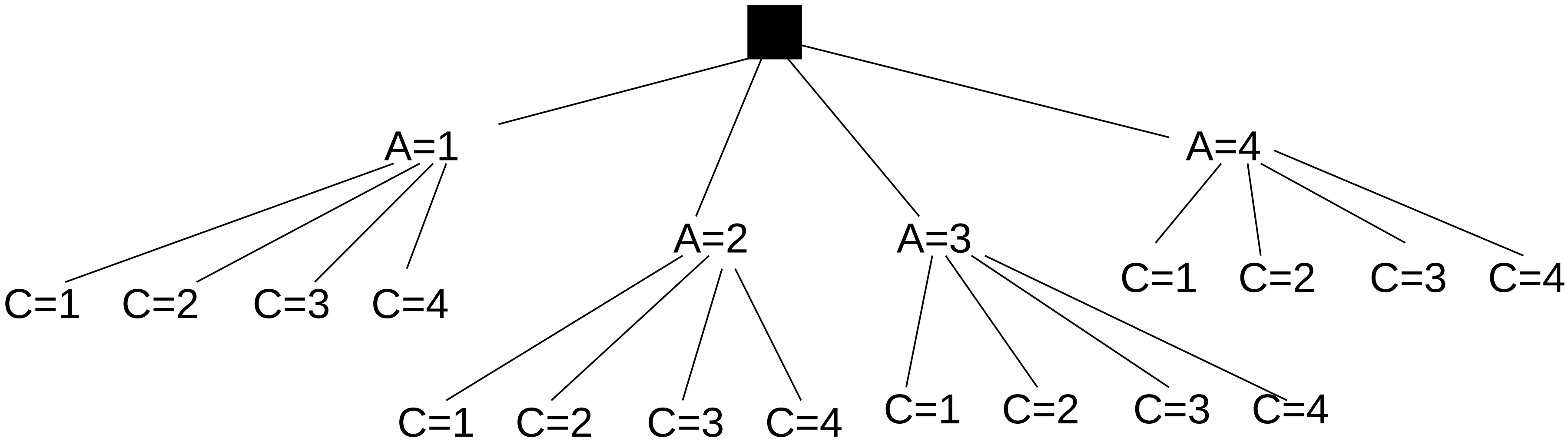
Solving CSPs by DFS: Example Efficiency

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: if we have the constraint $B \neq 3$, then **D_B** = {1, 2, 3, 4} _____________ domain consistent.

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.

Note: strictly speaking, when all of the constraints are binary, constraint nodes are not necessary: we can drop constraint nodes and use edges to indicate that a constraint holds between a pair of variables. However, **in this course we will always show constraint nodes.**

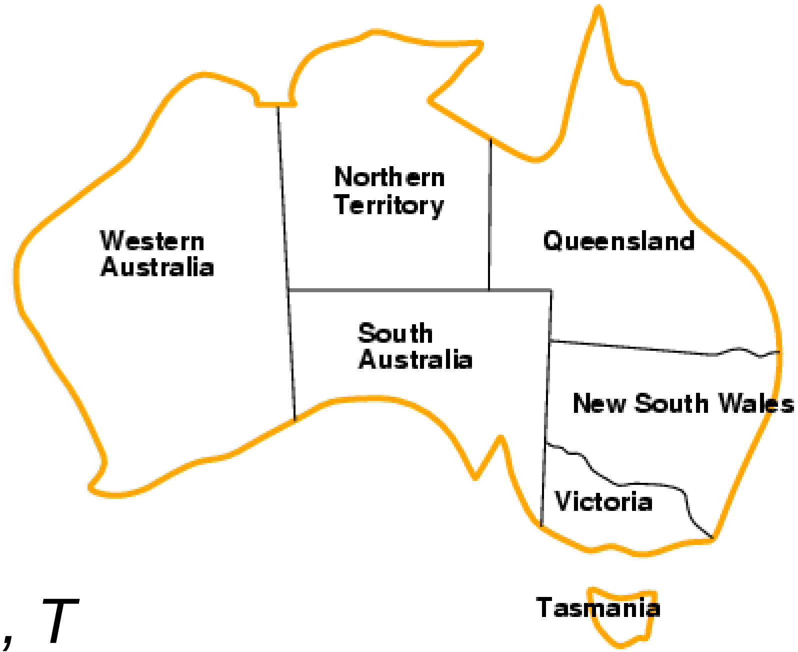
Example Constraint Network

Recall Example:

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

What would a constraint network for this example look like?

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

Group activity: draw a constraint network for this example

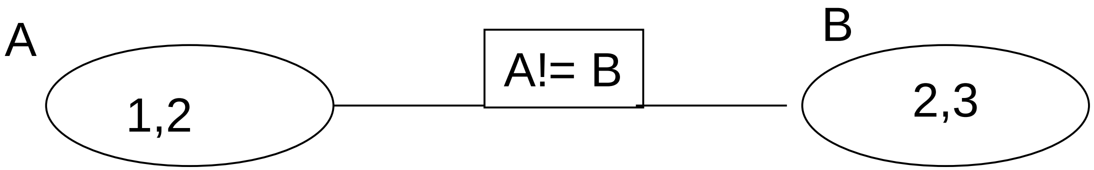
Lecture Overview

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

Arc Consistency

Definition (arc consistency)

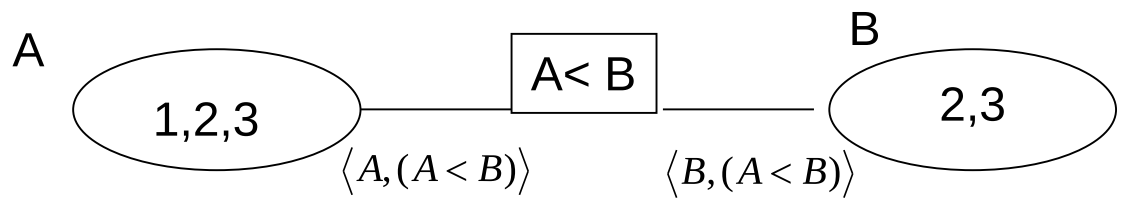
An arc $\langle X, r(X,Y)\rangle$ is arc consistent if for each value x in $dom(X)$ there is some value y in $dom(Y)$ such that $r(x,y)$ is satisfied.



Arc Consistency

Definition (arc consistency)

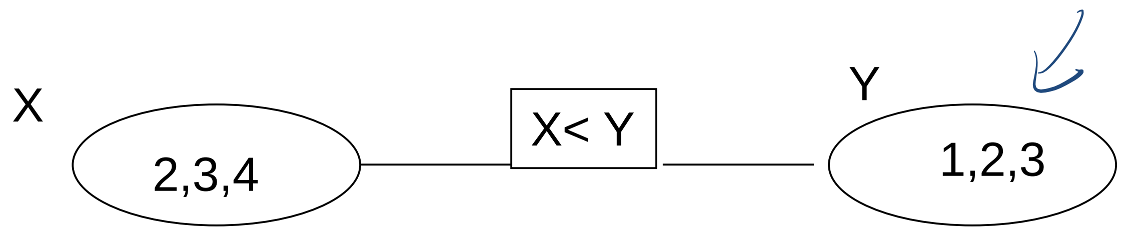
An arc $\langle X, r(X,Y) \rangle$ is arc consistent if for each value x in $dom(X)$ there is some value y in $dom(Y)$ such that $r(x,y)$ is satisfied.



A. Both arcs are consistent
B. Left consistent, right inconsistent
C. Right consistent, left inconsistent
D. Both arcs are inconsistent
E. I consistently fall asleep in this class

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.