CSPs: Arc Consistency & Domain Splitting

Computer Science cpsc322, Lecture 13

(Textbook Chpt 4.5 ,4.8)

February, 02, 2009

Lecture Overview

- Recap (CSP as search & Constraint Networks)
- Arc Consistency Algorithm
- Domain splitting

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

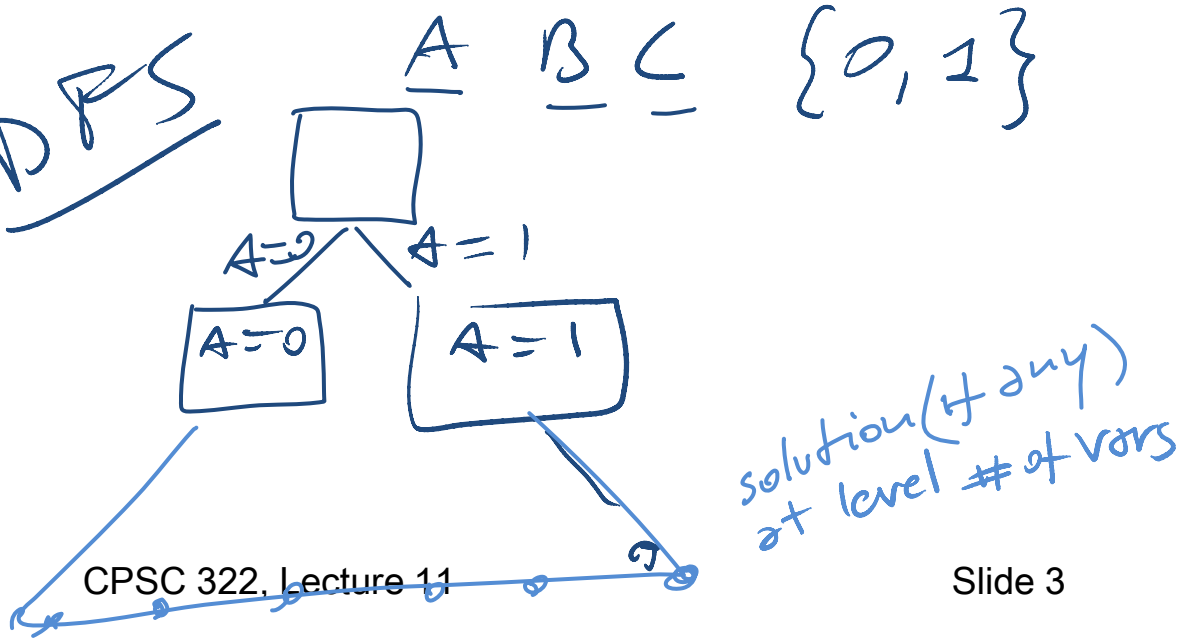
- State: assignments of values to a subset of the variables
- Successor function: assign values to a "free" variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: *none (all solutions at the same distance from start)*

Planning :

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

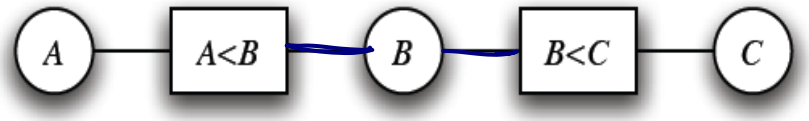
Inference

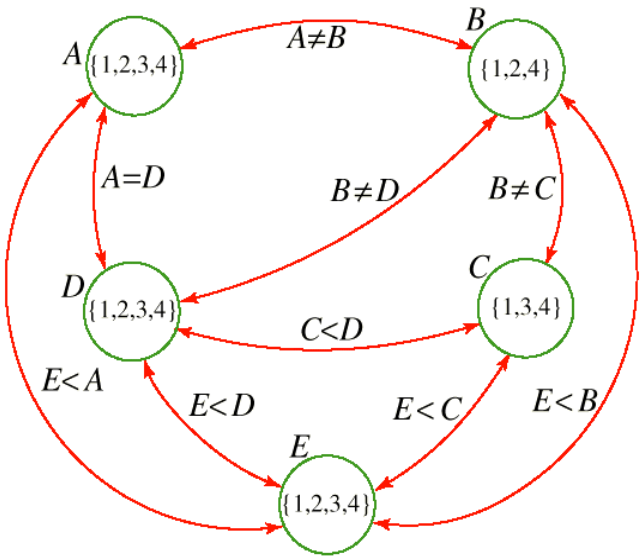
- State
- Successor function
- Goal test
- Solution
- Heuristic function



Recap: We can do much better..

- Build a constraint network:





- Enforce domain and arc consistency

Lecture Overview

- Recap
- Arc Consistency Algorithm
  - Abstract strategy
  - Details
  - Complexity
  - Interpreting the output
- Domain Splitting

Arc Consistency Algorithm: high level strategy

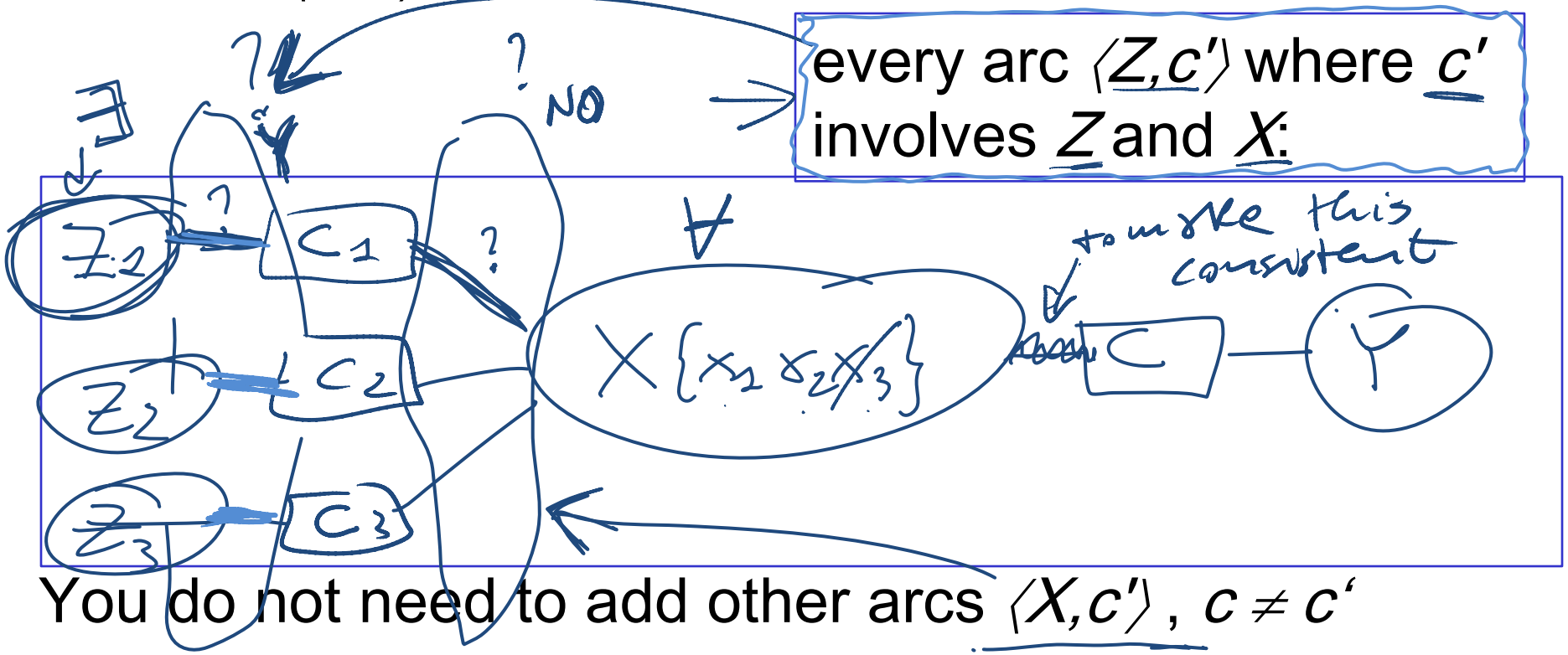
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited whenever….

- NOTE - Regardless of the order in which arcs are considered, we will terminate with the same result: an arc consistent network.

What arcs need to be revisited?

When we reduce the domain of a variable X to make an arc $\langle X,c \rangle$ arc consistent, we add……

every arc $\langle Z,c' \rangle$ where c' involves Z and X:



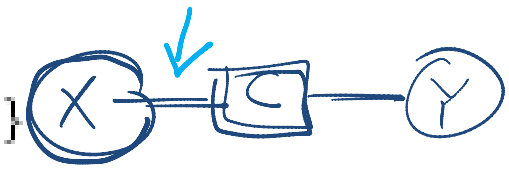
You do not need to add other arcs $\langle X,c' \rangle$, $c \neq c'$

- If an arc $\langle X,c' \rangle$ was arc consistent before, it will still be arc consistent (in the ``for all'' we'll just check fewer values)

Arc Consistency Algorithm (for binary C)

from text book

2: **Inputs**
3: V: a set of variables
4: dom: a function such that $dom(X)$ is the domain of variable X
5: C: set of constraints to be satisfied
6: **Output**
7: arc consistent domains for each variable
8: **Local**
9: D_X is a set of values for each variable X
10: TDA is a set of arcs — to do arcs
11: **for each** variable X **do**
12: $D_X \leftarrow dom(X)$
13: $TDA \leftarrow \{\langle X, c\rangle \mid c \in C \text{ and } X \in scope(c)\}$ — all the arcs



14: **end for each**
15: **while** $TDA \neq \{\}$ **do** — to be checked
16: **select** $\langle X, c\rangle \in TDA$;
17: $TDA \leftarrow TDA \setminus \{\langle X, c\rangle\}$;
18: $ND_X \leftarrow \{x \mid x \in D_X \text{ and some } \{X = x, Y_1 = y_1, \ldots, Y_k = y_k\} \in c \text{ where } y_i \in D_{Y_i} \text{ for all } i\}$ — $y \in DY$ (x,y) satisfies c
19: **if** $ND_X \neq D_X$ **then** — arc inconsistent
20: $TDA \leftarrow TDA \cup \{\langle Z, c'\rangle \mid X \in scope(c'), c' \text{ is not } c, Z \in scope(c') \setminus \{X\}\}$ — adds arcs to TDA (see prev. slide)
21: $D_X \leftarrow ND_X$ — reducing X
22: **end if**

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS d^n)
  - let the max size of a variable domain be d
  - let the number of variables be n
  - The max number of binary constraints is $n(n-1)/2$
- How many times the same arc can be inserted in the ToDoArc list?
  - d + d = $2d$
- How many steps are involved in checking the consistency of an arc?
  - d^2

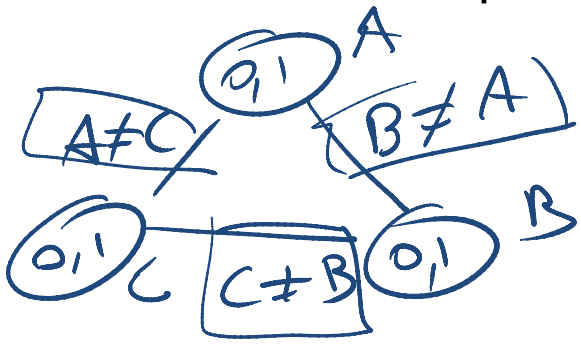
much better

$O(n(n-1)d^3)$

OVERALL COMPLEXITY

Arc Consistency Algorithm: Interpreting Outcomes

- Three possible outcomes (when all arcs are arc consistent):
  - One domain is empty → no solution
  - Each domain has a single value → unique solution :)
  - Some domains have more than one value → may or may not be a solution
    - in this case, arc consistency isn't enough to solve the problem: we need to perform search



example of arc consistent Const. Net with no solution

Lecture Overview

- Recap
- Arc Consistency
- Domain splitting

Domain splitting (or case analysis)

- Arc consistency ends: Some domains have more than one value → may or may not be a solution
  - A. Apply Depth-First Search with Pruning
  - B. Split the problem in a number of disjoint cases

$CSP = \{X = \{0, 1, 2, 3\} \dots\}$

$CSP_1 = \{X = \{0, 1\} \dots\}$ $CSP_2 = \{X = \{2, 3\} \dots\}$

- Set of all solution equals to….

$$sol(CSP) = \bigcup_i sol(CSP_i)$$

But what is the advantage?

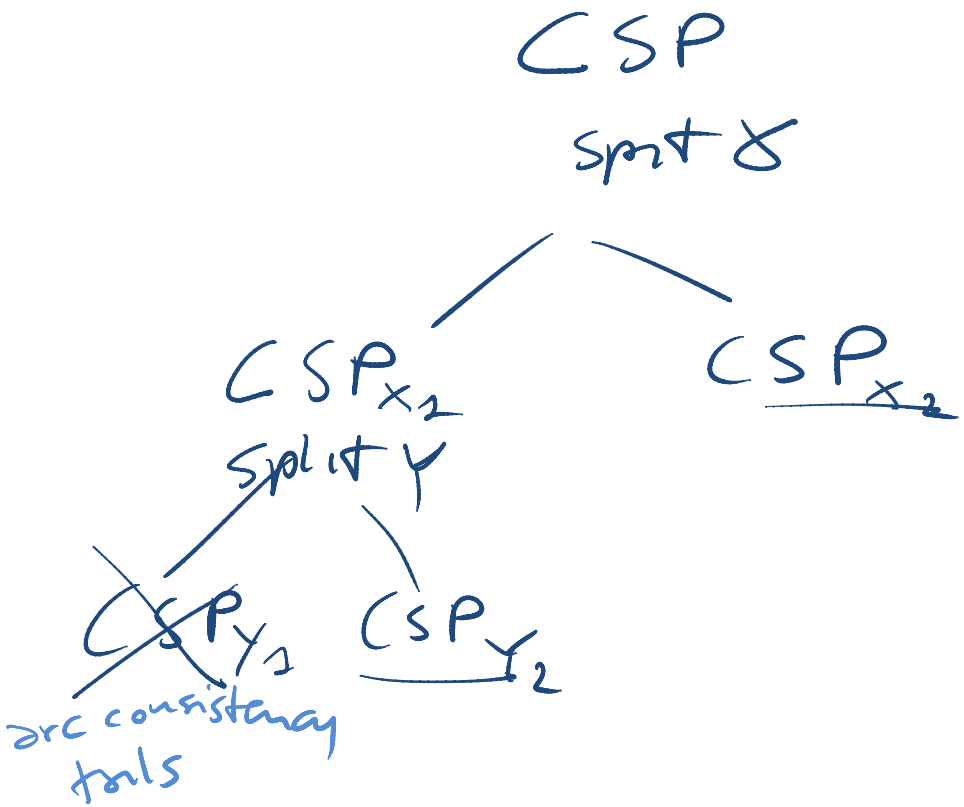
We may be able to apply Arc Consist. again

- Simplify the problem using **arc consistency**
- No unique solution i.e., for at least one var, |dom(X)|>1
- **Split X** → two or more subsets S_i $\bigcap_i S_i = \emptyset$ $\bigcup_i S_i = dom(X)$ $CSP_2\{X=\{0\}\}$ $CSP_1\{X=\{2,3\}\}$
- For all the splits
  - Restart arc consistency on arcs <Z, r(Z,X)>

these are the ones that are possibly……inconsistent

- Disadvantage ☹: you need to keep all these CSPs around (vs. lean states of DFS)

Searching by domain splitting



- Disadvantage ☹: you need to keep all these CSPs around (vs. lean states of DFS)

Learning Goals for today's class

You can:

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

- Define/read/write/trace/debug domain splitting and its integration with arc consistency

Next Class

- **Local search:**
- Many search spaces for CSPs are simply too big for systematic search (but solutions are densely distributed).
  - Keep only the current state (or a few)
  - Use very little memory / often find reasonable solution

- ….. **Local search for CSPs**

K-ary vs. binary constraints

- **Not a topic for this course** but if you are curious about it…
- Wikipedia example clarifies basic idea…
- http://en.wikipedia.org/wiki/Constraint_satisfaction_dual_problem
- The **dual problem** is a reformulation of a constraint satisfaction problem expressing each constraint of the original problem as a variable. Dual problems only contain binary constraints, and are therefore solvable by algorithms tailored for such problems.
- See also: **hidden transformations**