CSPs: Arc Consistency & Domain Splitting

Computer Science cpsc322, Lecture 13

(Textbook Chpt 4.5 ,4.6)

February, 01, 2010

Lecture Overview

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

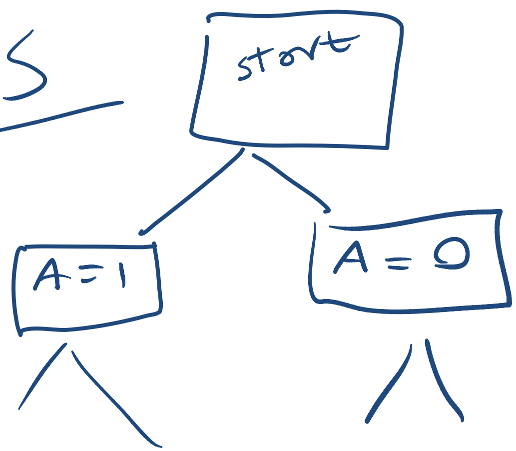
Standard Search vs. Specific R&R systems

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)*

A B {0 1}

DFS



solutions at level # of vars

Planning :

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

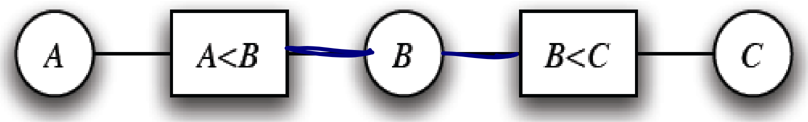
Query

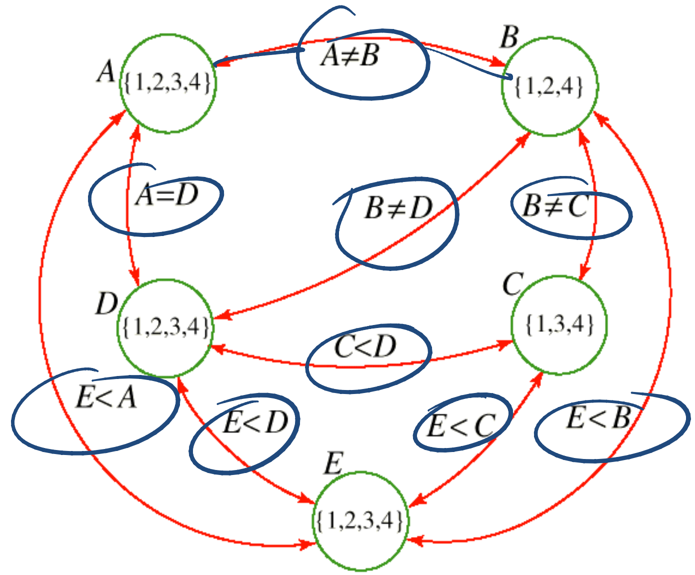
- State
- Successor function
- Goal test
- Solution
- Heuristic function

Recap: We can do much better..

- Build a constraint network:

A B C   A < B   B < C





- Enforce domain and arc consistency

AIspace

Lecture Overview

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

Arc Consistency Algorithm: high level strategy

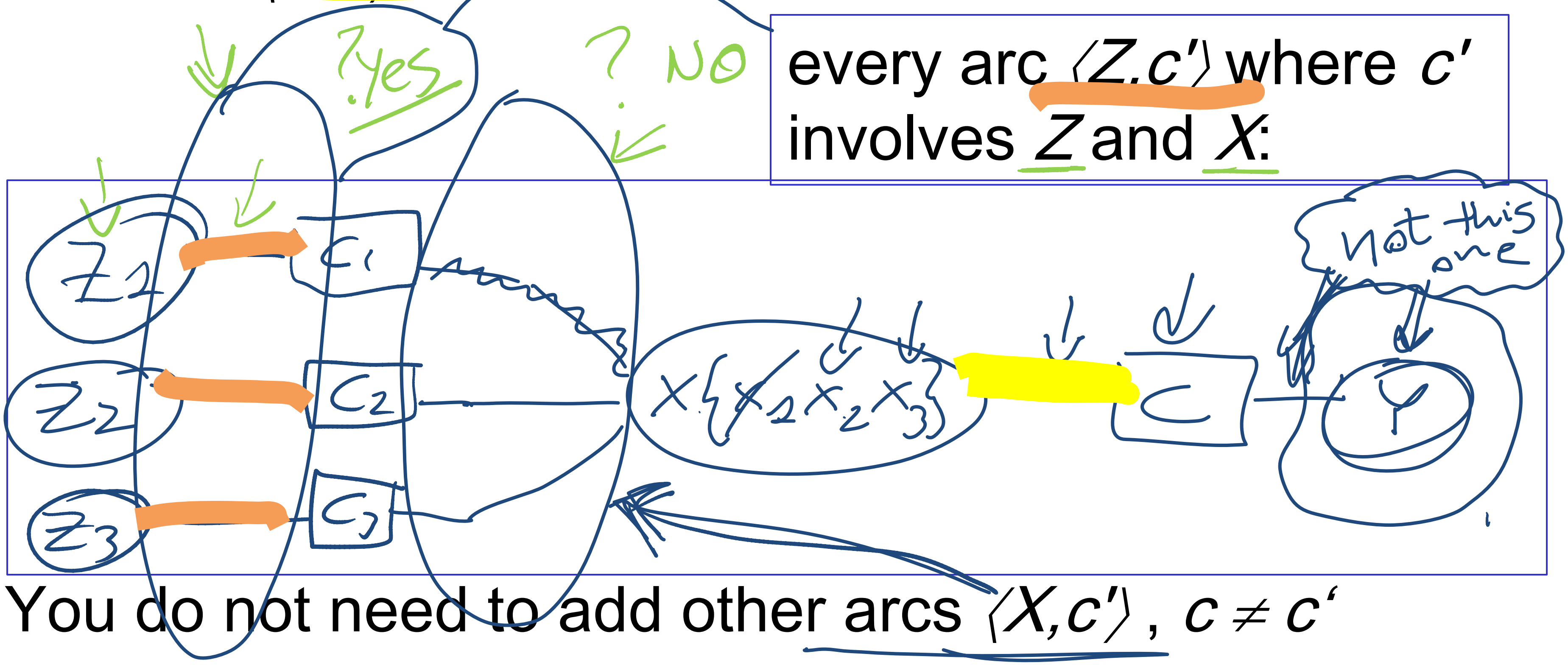
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited whenever….

- NOTE - Regardless of the order in which arcs are considered, we will terminate with the same result

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)

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

        D_X is a set of values for each variable X

        TDA is a set of arcs   TO DO ARC LIST

    for each variable X **do**   INITIALIZE

        $D_X \leftarrow dom(X)$

        TDA ←{⟨X,c⟩| c∈C and X∈scope(c)}   BINARY

    while (TDA ≠{})

        select ⟨X,c⟩ ∈TDA;   arc TO BE CHECKED

        TDA ←TDA \ {⟨X,c⟩};

        $ND_X \leftarrow \{x \mid x \in D_X$ and there is $y_i \in D_{Y_i}$ $\{X=x, Y=y\} \in c$ };   satisfies C

        if ($ND_X \neq D_X$) **then**   arc is not consistent

            TDA ←TDA ∪{⟨Z,c'⟩|X ∈scope(c'), c' is not c, Z∈scope(c') \ {X} }   add arcs to TDA see previous slide

            $D_X \leftarrow ND_X$ reduce the domain

    return {D_X| X is a variable}

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

$$\boxed{O(d^3 n^2)}$$

- How many steps are involved in checking the consistency of an arc? d^2

$\{x_1 \ldots x_d\}$ $\{y_1 \ldots y_d\}$

OVERALL COMPLEXITY

Arc Consistency Algorithm: Interpreting Outcomes

- Three possible outcomes (when all arcs are arc consistent):
  - One domain is empty → no sol
  - Each domain has a single value → unique sol
  - 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

see arc consistency (AC) practice exercise

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

$CSP = \{ X = \{x_1\ x_2\ x_3\ x_4\} \ldots \}$

$CSP_1 \{ X = \{x_1 x_2\} \ldots \}$ $\quad$ $CSP_2 \{ X \{x_3\ x_4\} \}$

- Set of all solution equals to….

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

But what is the advantage?

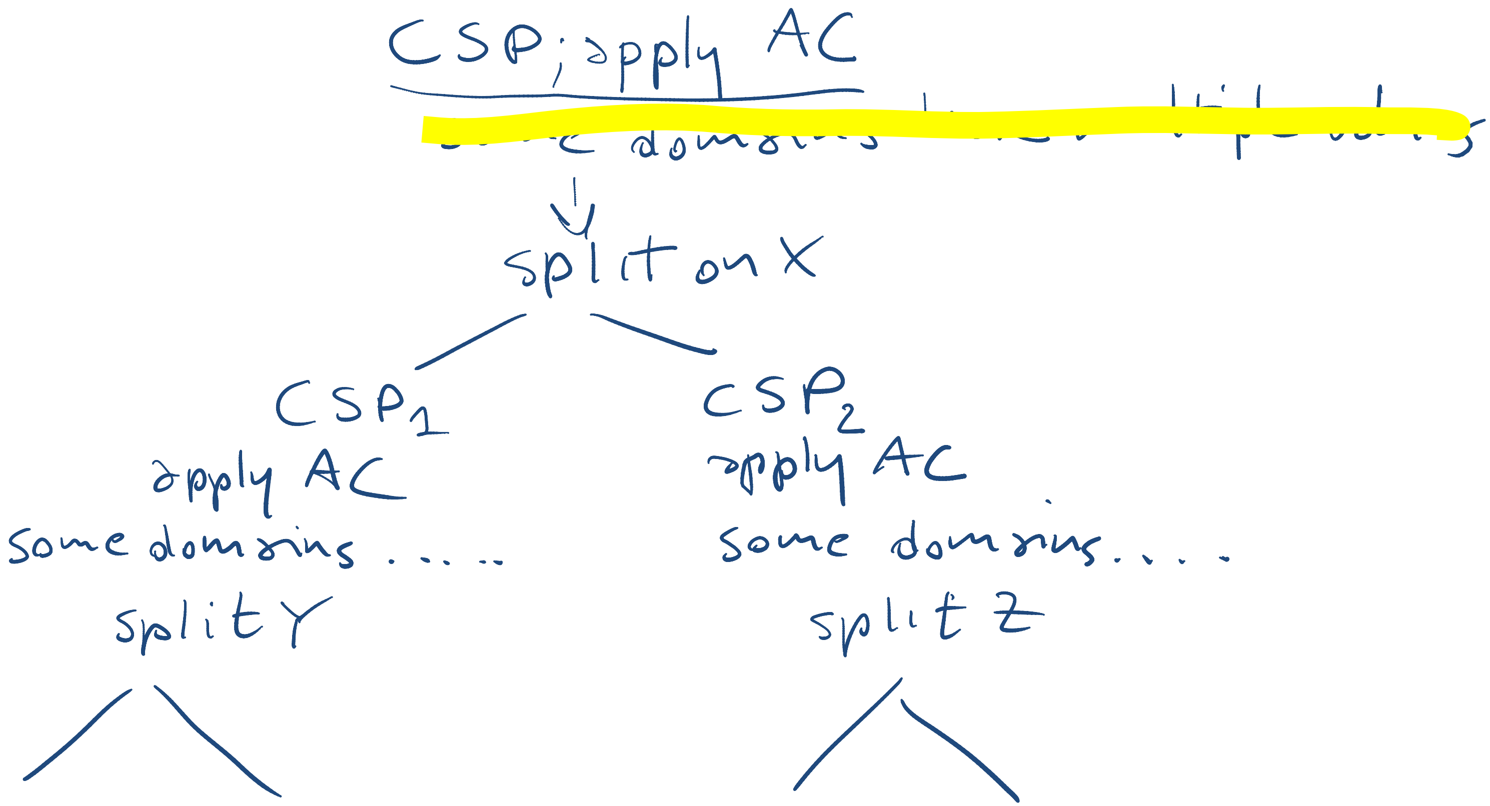
By reducing dom(X) we may be able to…. run AC again

- Simplify the problem using **arc consistency**
- No unique solution i.e., for at least one var, |dom(X)|>1
- **Split X**
- For all the splits
  - Restart arc consistency on arcs <Z, r(Z,X)> Initial TDA

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 (Chpt. 4.8)

- **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**