CSPs: Arc Consistency

& Domain Splitting

Computer Science cpsc322, Lecture 13

(Textbook Chpt 4.5 ,4.6)

May, 30, 2017

Lecture Overview

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

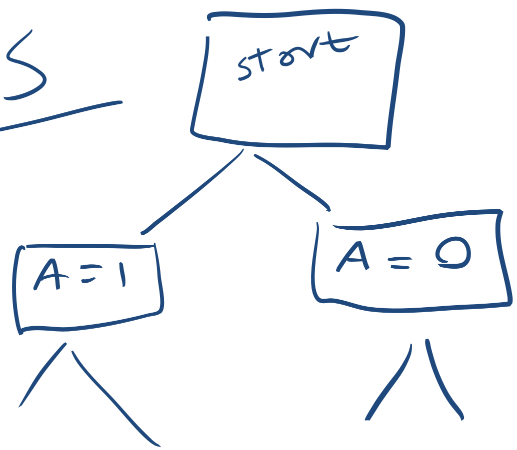
Standard Search vs. Specific R&R systems

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)*

DFS

A B {0 1}



solutions at level # of vars

Planning :

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

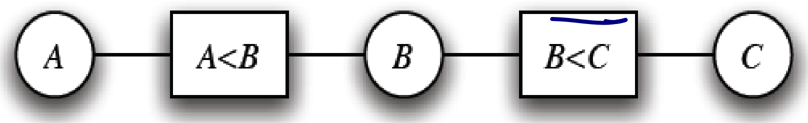
Query

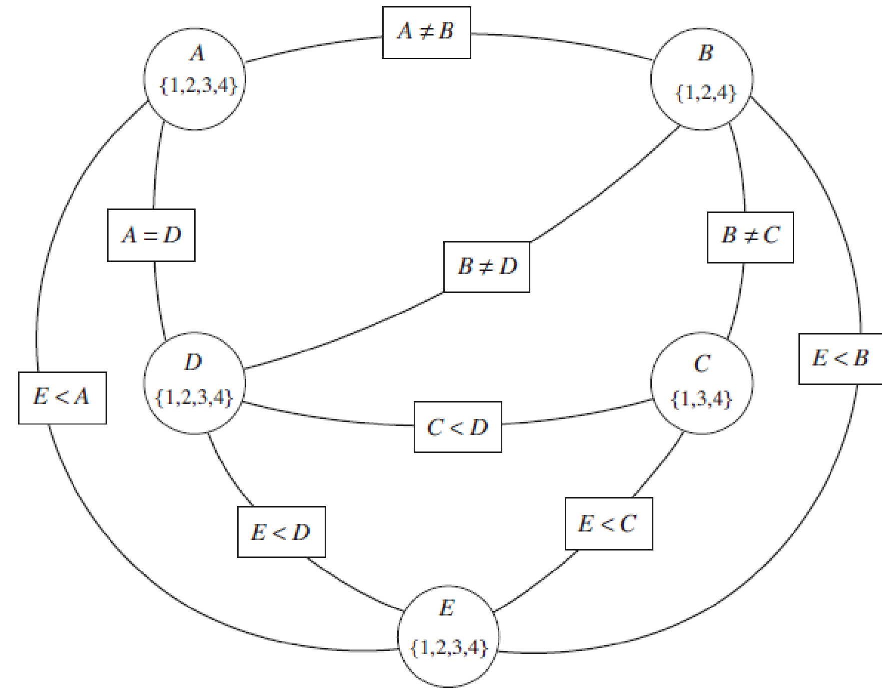
- State
- Successor function
- Goal test
- Solution
- Heuristic function

Recap: We can do much better..

- Build a constraint network:

A B C   A< B   B < C





- Enforce domain and arc consistency

Lecture Overview

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

Arc Consistency Algorithm: high level strategy

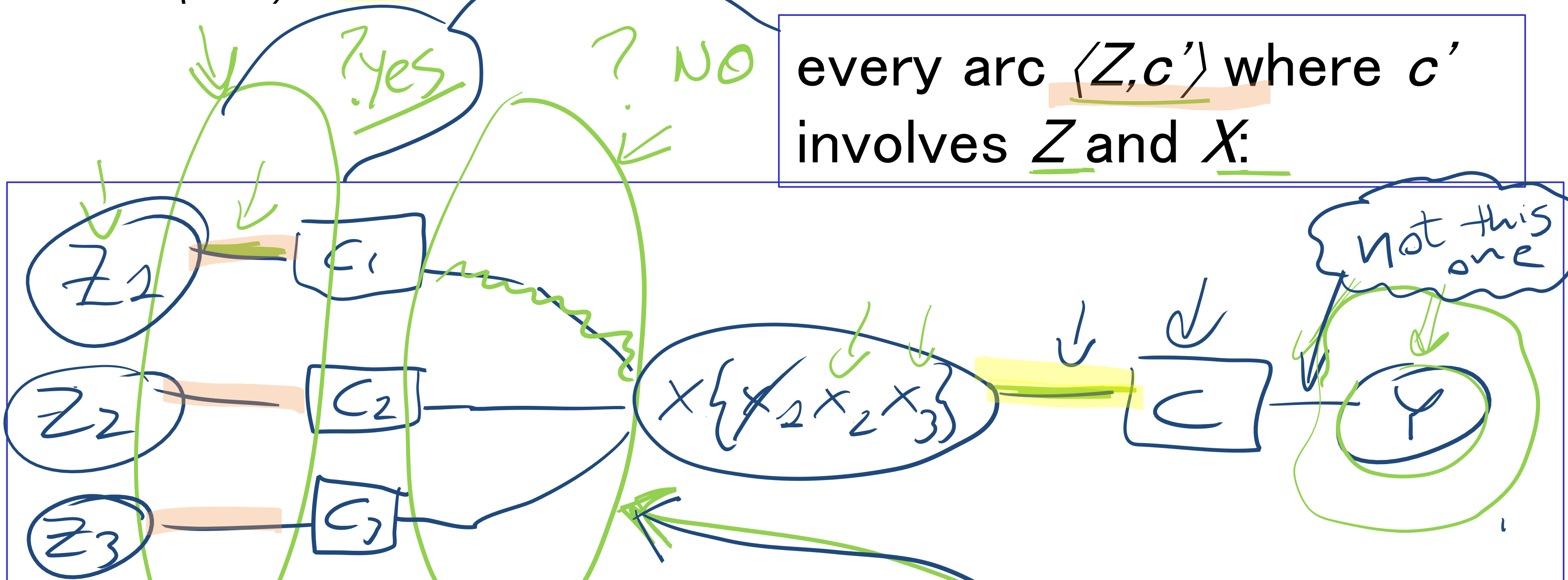
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited whenever….

- 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 PSEUDO-CODE

TDA ← all arcs in Constraint Network

WHILE (TDA is not empty)
- select arc a from TDA
  IF (a is not consistent) THEN
  - make a consistent
  - add arcs to TDA that may now be inconsistent

SEE PREVIOUS SLIDE

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

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

TDA: ToDoArcs, blue arcs in AIspace

for each variable X **do**

  $\mathbf{D}_X \leftarrow \mathrm{dom}(X)$

TDA $\leftarrow \{\langle X,c\rangle \mid X \in V,\ c \in C$ and $X \in \mathrm{scope}(c)\}$

while (TDA $\neq \{\}$)

  select $\langle X,c\rangle \in$ TDA

  TDA $\leftarrow$ TDA $\setminus \{\langle X,c\rangle\}$

  $ND_X \leftarrow \{x \mid x \in \mathbf{D}_X$ and $\exists\, y \in \mathbf{D}_Y$ s.t. (x, y) satisfies c$\}$

  if $(ND_X \neq \mathbf{D}_X)$ **then**

    TDA $\leftarrow$ TDA $\cup \{ \langle Z,c'\rangle \mid X \in \mathrm{scope}(c'),\ c' \neq c,\ Z \in \mathrm{scope}(c') \setminus \{X\} \}$

    $\mathbf{D}_X \leftarrow ND_X$

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

X's domain changed: ⇒ arcs (Z,c') for variables Z sharing a constraint c' with X are added to TDA

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

If arc was inconsistent

Domain is reduced

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 ?

A. n * d

B. d * d

C. (n * (n-1)) / 2

D. (n * d) / 2

iclicker.

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS)
  - let the max size of a variable domain be ***d***
  - let the number of variables be ***n***
- How many times the same arc can be inserted in the ToDoArc list?

A. **n** B. **d** C. **n * d** D. **d^2**

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

A. **n^2** B. **d** C. **n * d** D. **d^2**

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 (two) disjoint cases

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

$$CSP_1 \{ X = \{x_1 x_2\} \ldots \} \qquad CSP_2 \{ X \{x_{3\,4}\} \}$$

- 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

CSP; apply AC

some domains have multiple values

split on X

CSP$_1$

apply AC

some domains ....

split Y

CSP$_2$

apply AC

some domains....

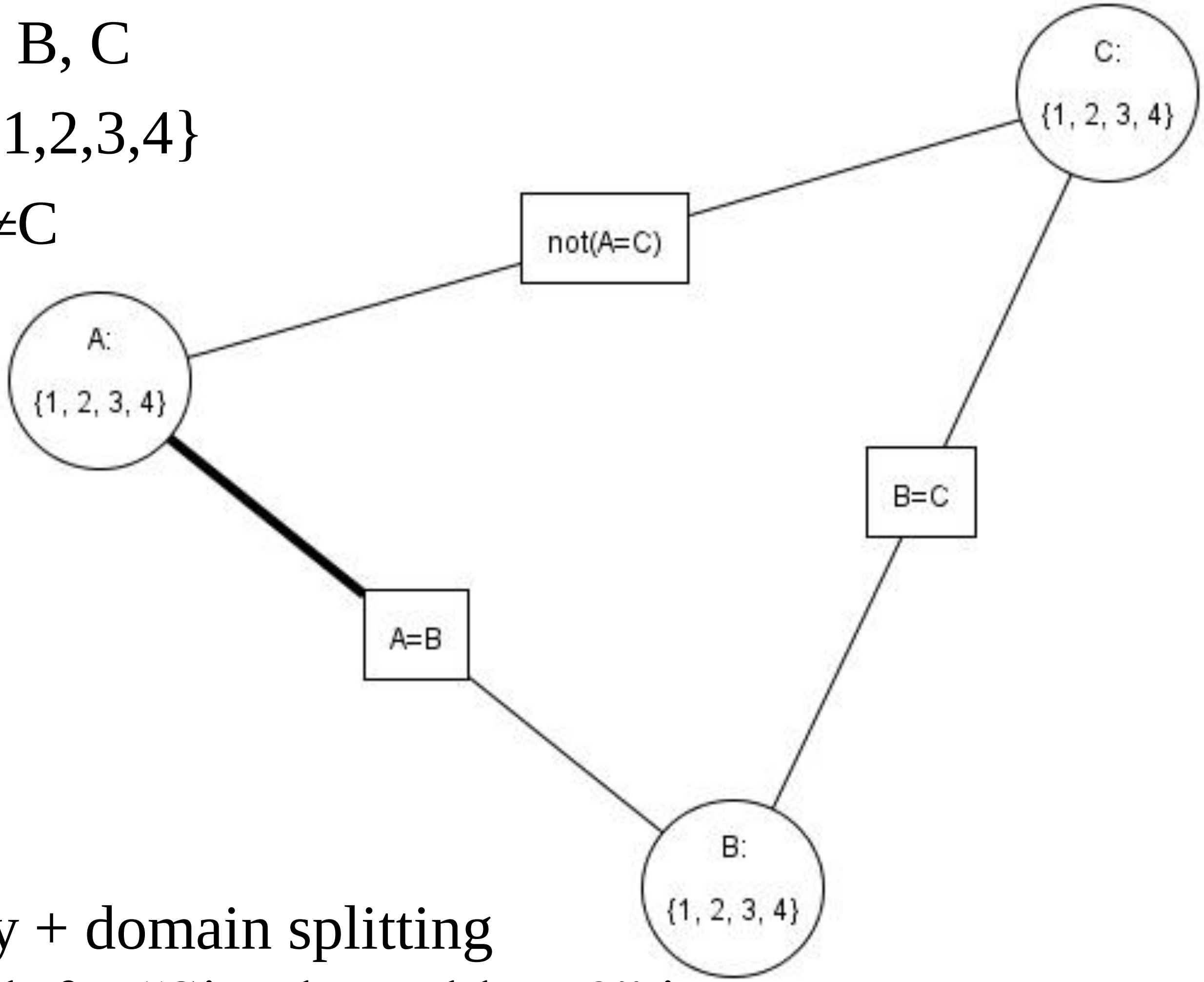
split Z

More formally: Arc consistency with domain splitting as another formulation of CSP as search

- **Start state**: run AC on vector of original domains ($dom(V_1)$, …, $dom(V_n)$)
- **States:** "remaining" domains ($D(V_1)$, …, $D(V_n)$) for the vars with $D(V_i) \subseteq dom(V_i)$ for each V_i
- **Successor function:**
  - split one of the domains + run arc consistency
- **Goal state**: vector of unary domains that satisfies all constraints
  - That is, only one value left for each variable
  - The assignment of each variable to its single value is a model
- **Solution**: that assignment

Domain Splitting in Action:

- 3 variables: A, B, C
- Domains: all {1,2,3,4}
- A=B, B=C, A≠C



- Let's trace
  arc consistency + domain splitting
  for this network for "Simple Problem 2" in AIspace

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

- **Work on CSP Practice Ex:**
  - Exercise 4.A: arc consistency
  - Exercise 4.B: constraint satisfaction problems
  -

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**