CSPs: Arc Consistency & Domain Splitting

Computer Science cpsc322, Lecture 13

(Textbook Chpt 4.5 ,4.6)

Oct, 04, 2013

Lecture Overview

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

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

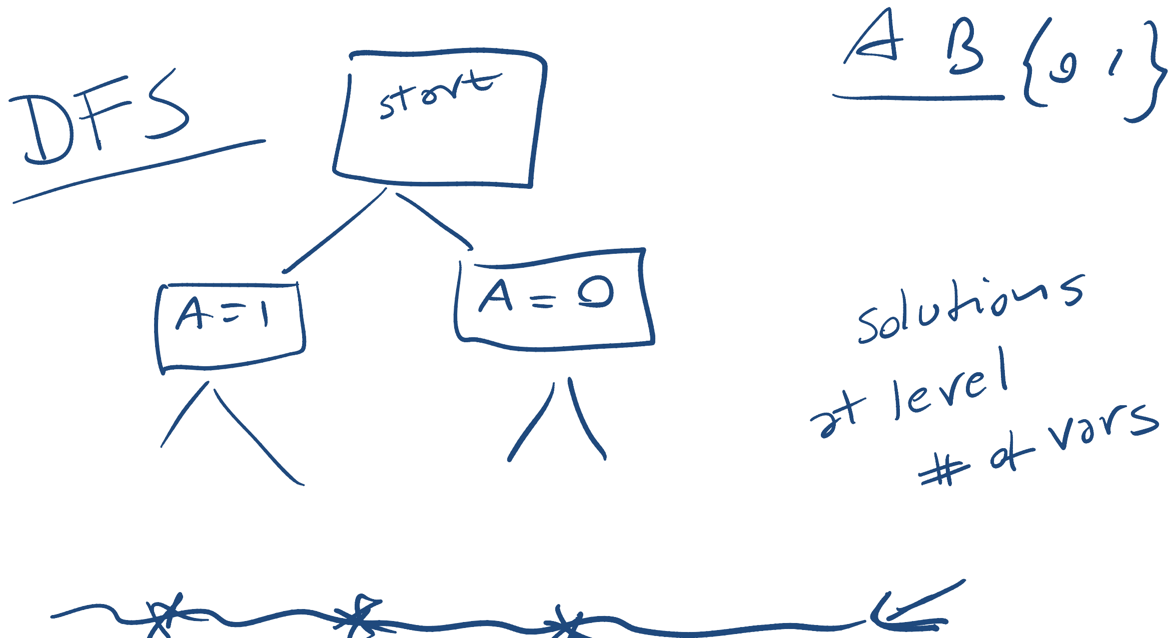
- State: assignments of values to a subset of the variables
- Successor function: assign values to a "free" variable
- Goal test: set of constraints
- Solution: possible world that satisfies the constraints
- Heuristic function: *none (all solutions at the same distance from start)*

Planning :

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

Query

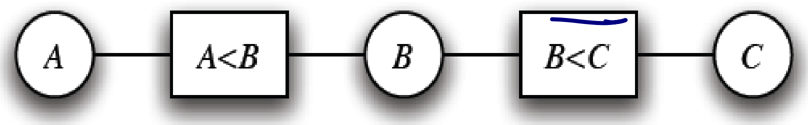
- State
- Successor function
- Goal test
- Solution
- Heuristic function

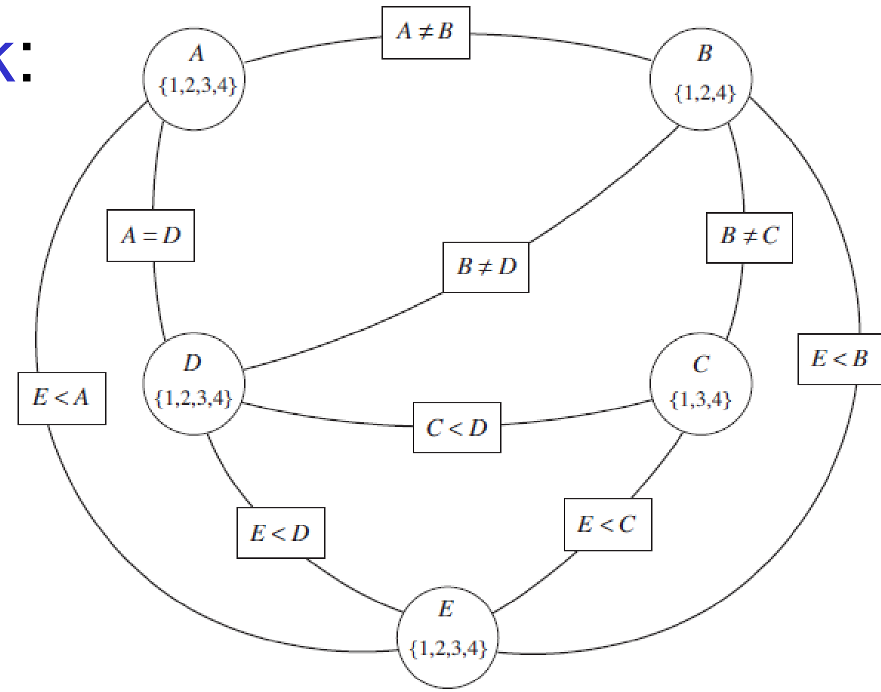


Recap: We can do much better..

- 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

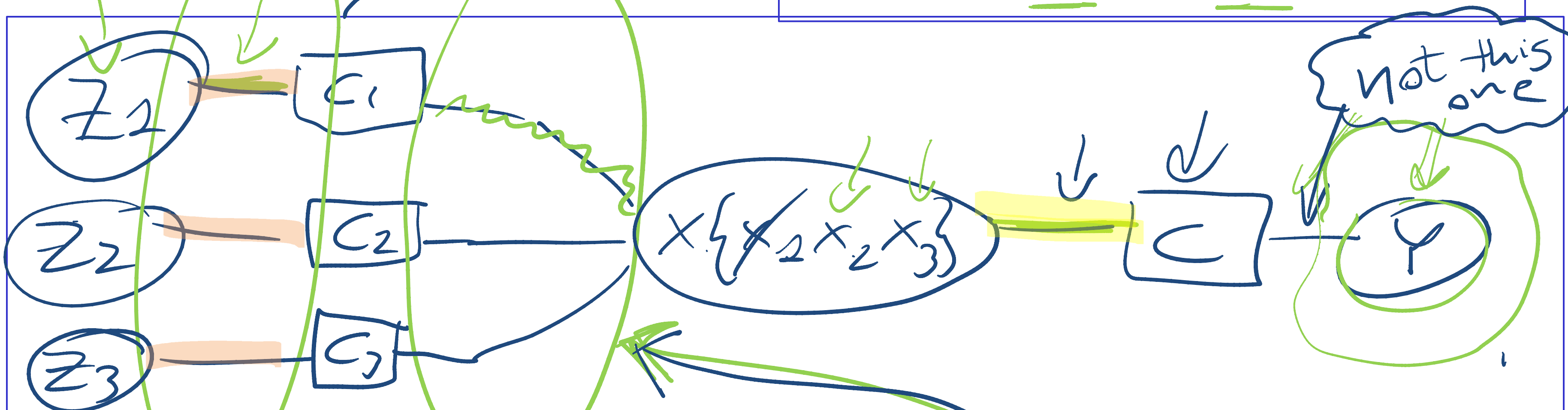
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited whenever….

- 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

```
for each variable X do
    D_X ← dom(X)
TDA ← {⟨X,c⟩ | X ∈ V, c ∈ C and X ∈ scope(c)}

while (TDA ≠ {})
    select ⟨X,c⟩ ∈ TDA
    TDA ← TDA \ {⟨X,c⟩}
    ND_X ← {x | x ∈ D_X and ∃ y ∈ D_Y s.t. (x, y) satisfies c}
    if (ND_X ≠ D_X) then
        TDA ← TDA ∪ { ⟨Z,c'⟩ | X ∈ scope(c'), c' ≠ c, Z ∈ scope(c') \ {X} }
        D_X ← ND_X

return {D_X | X is a variable}
```

TDA: ToDoArcs, blue arcs in AIspace

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

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

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

If arc was inconsistent

Domain is reduced

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS d^4)
  - 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

$O(d^3 n^2)$

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

$\{X_1 \dots X_d\}$ $\{Y_1 \dots Y_d\}$

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

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

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

- Set of all solution equals to….

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

But what is the advantage?

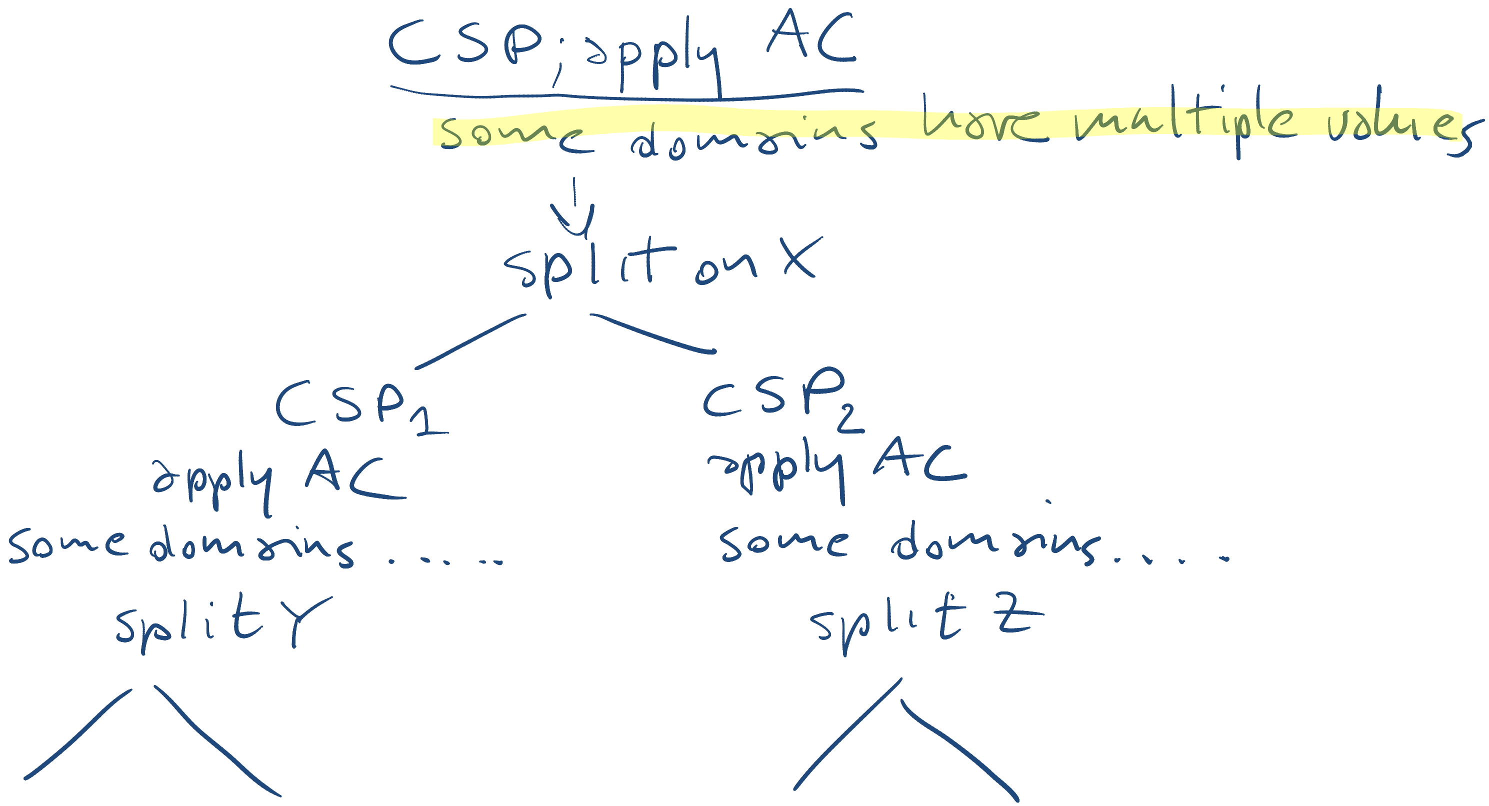
By reducing dom(X) we may be able to….

- 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)>

these are the ones that are possibly………..

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

Searching by domain splitting



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

- **States:** "remaining" domains ($D(V_1)$, …, $D(V_n)$)for the vars with $D(V_i) \subseteq dom(V_i)$ for each V_i
- **Start state**: run AC on vector of original domains ($dom(V_1)$, …, $dom(V_n)$)
- **Successor function:**
  - reduce 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

Searching by domain splitting

Apply AC to original domains

If domains with multiple values

Split on one

apply AC to remaining domains

If domains with multiple values
Split on one

apply AC to remaining domains

If domains with multiple values.....Split on one

How many CSPs do we need to keep around at a time?
Assume **solution at depth m** and **2 children at each split**

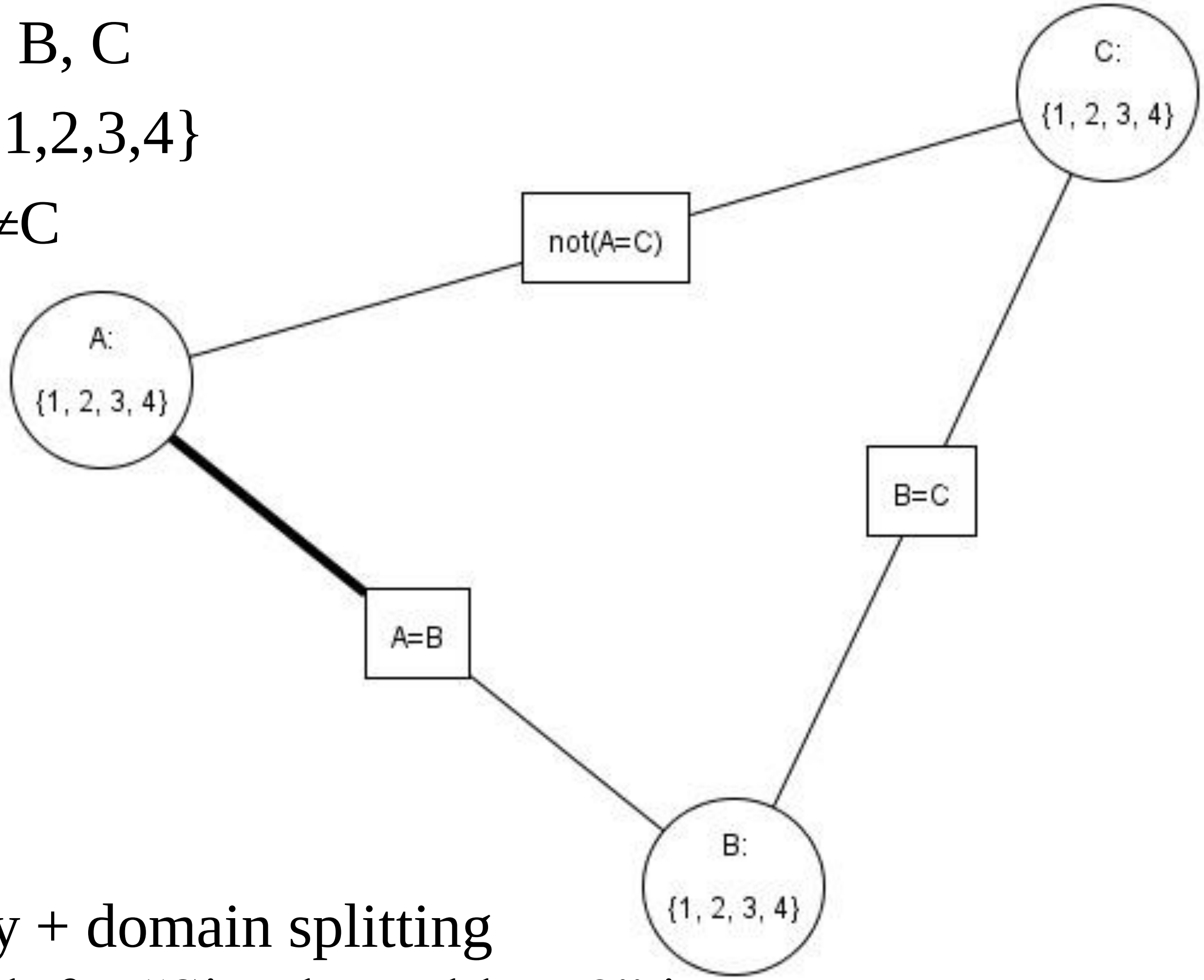
A. **It depends** B. **2m** C. **m^2** D. **2^m**

K-ary vs. binary constraints

- **<u>Not a topic for this course</u>** 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**

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**