CSPs: Arc Consistency & Domain Splitting

CPSC 322 Lecture 12

Lecture Overview

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

Standard Search vs. Specific R&R

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 :

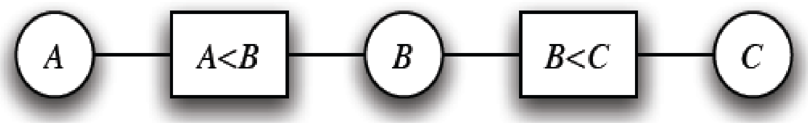
- State
- Successor function
- Goal test
- Solution
- Heuristic function

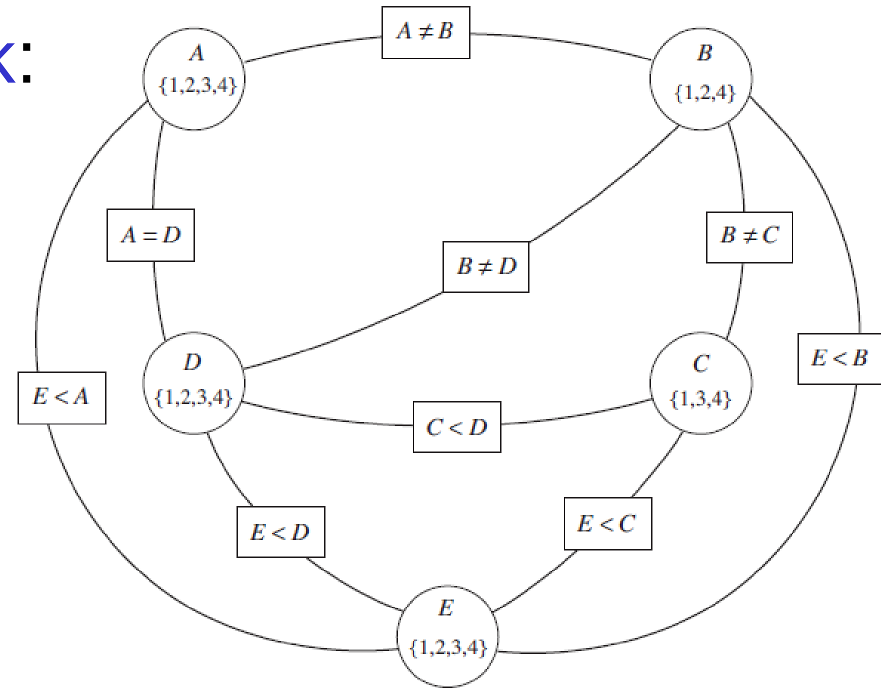
Query

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

Recap: We can do much better..

- Build a constraint network:





- Enforce domain and arc consistency

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

Lecture Overview

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

Arc Consistency Algorithm: high level strategy

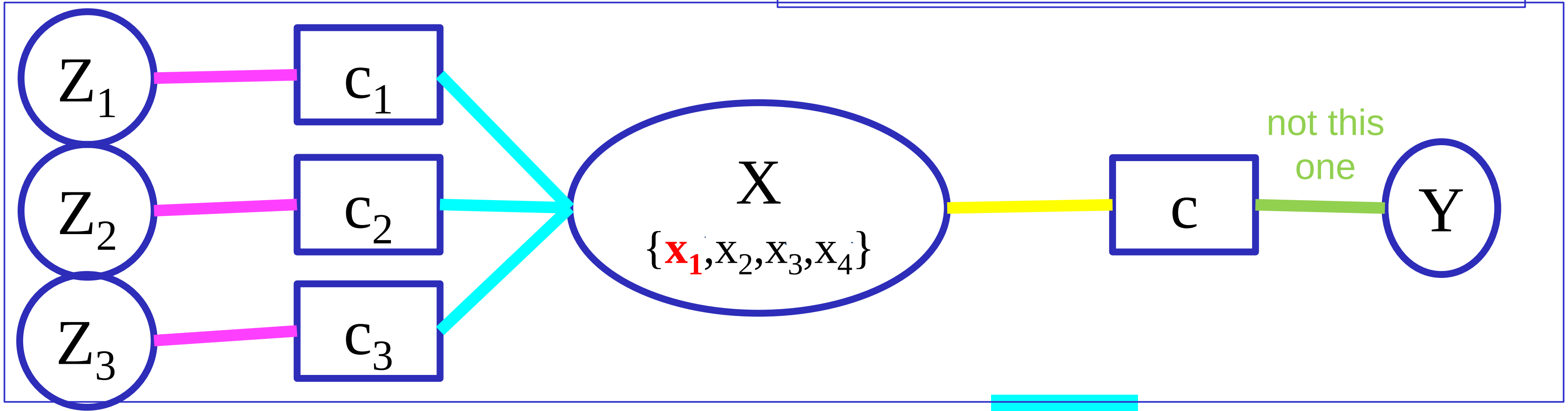
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited when….?

- 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 ⟨*X*,*c*⟩ arc consistent, **we add**……

every arc ⟨*Z*,*c'*⟩ where *c'* involves *Z* and *X*:



You do not need to add other arcs ⟨*X*,*c'*⟩ , $c \neq c'$

- If an arc ⟨*X*,*c'*⟩ was arc consistent before, it will still be arc consistent (in the "for all" we'll just check fewer values)

Arc Consistency Pseudocode

```
TDA ← all arcs in constraint network
while TDA is not empty:
•  select arc a from TDA
•  if a is not consistent:
   •  make a consistent
   •  add arcs to TDA that may now be
      inconsistent
```

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**
    $\mathbf{D}_X \leftarrow \text{dom}(X)$
TDA $\leftarrow \{\langle X,c \rangle \mid X \in V, c \in C$ and $X \in \text{scope}(c)\}$

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 $\cup \{ \langle Z,c' \rangle \mid X \in \text{scope}(c'), c' \neq c, Z \in \text{scope}(c') \setminus \{X\} \}$
        $\mathbf{D}_X \leftarrow ND_X$

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

TDA: ToDoArcs, blue arcs in AIspace

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

X's domain changed: $\Rightarrow$ 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
  - let the max size of a variable domain be ***d***
  - let the number of variables be ***n***
  - Let all constraints be **binary**

  - The max number of binary constraints is ?

A. n * d

B. d * d

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

D. (n * d) / 2

Arc Consistency: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS, which is $\mathbf{d^n}$)
  - let the max size of a variable domain be ***d***
  - let the number of variables be ***n***
  - Let all constraints be **binary**

- How many times can the same arc be inserted in the ToDoArc list?

A. **n** B. **d** C. **n * d** D. $\mathbf{d^2}$

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

A. $\mathbf{n^2}$ B. **d** C. **n * d** D. $\mathbf{d^2}$

Arc Consistency Algorithm: Complexity

- Let's determine worst-case complexity of this procedure
  - let the max size of a variable domain be ***d***
  - let the number of variables be ***n***
  - The max number of binary constraints is ________

- How many times can the same arc be inserted in the ToDoArc list? __________
- How many steps are involved in checking the consistency of an arc? _________

- Overall complexity: $\boldsymbol{O(n^2d^3)}$

Arc Consistency Algorithm: Interpreting Outcomes

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

Lecture Overview

- **Recap**
- Arc Consistency
- Domain splitting

Domain splitting (or case analysis)

- Arc consistency ends: Some domains have more than one value $\rightarrow$ may or may not be a solution
  A. Apply Depth-First Search with Pruning
  B. Split the problem in a number of (eg. two) disjoint cases
     - The set of all solutions is….

But what is the advantage?

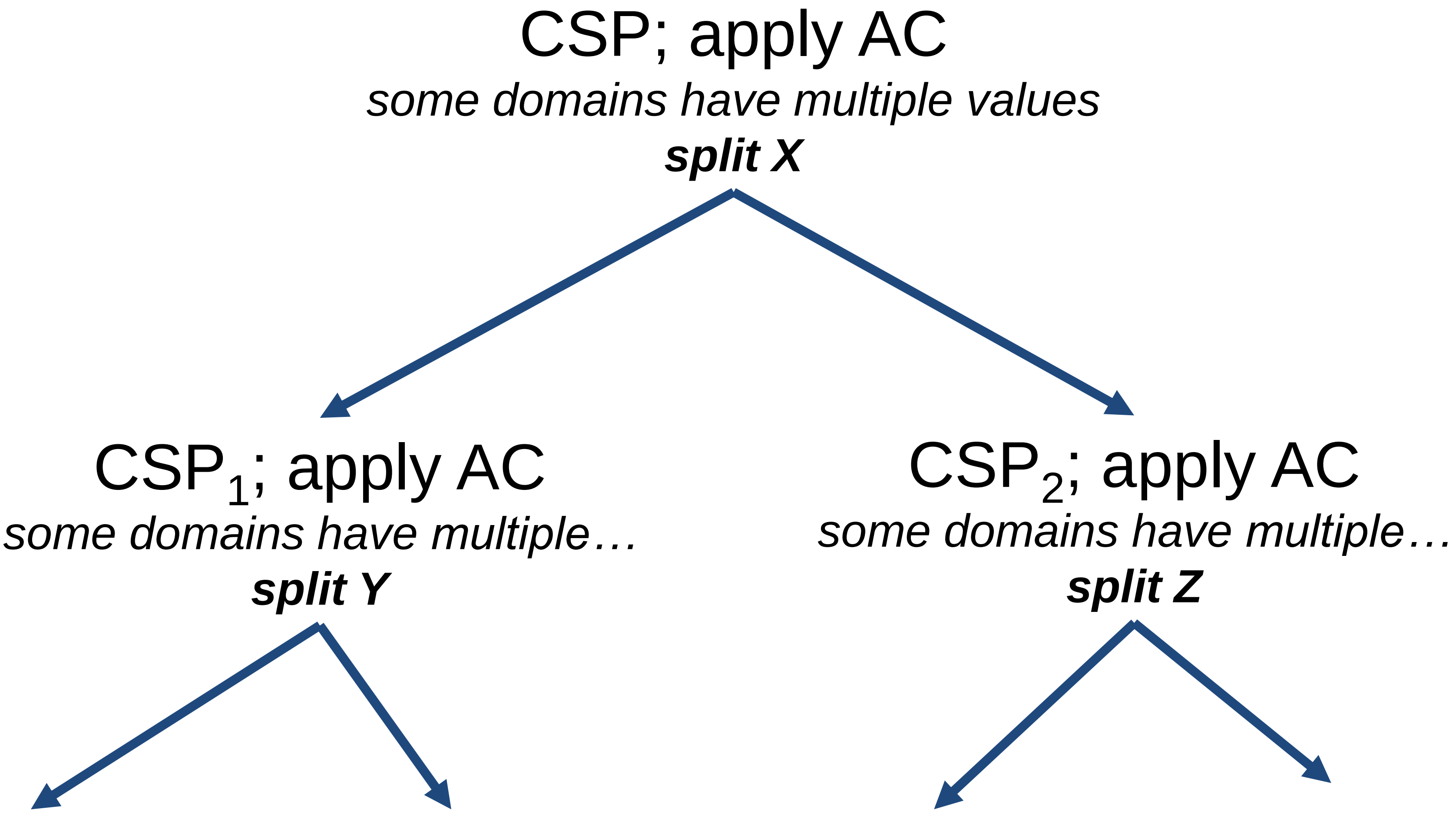
By reducing dom(X) we may be able to **run AC again**

- 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)>

these are the ones that are possibly **inconsistent**

- **Disadvantage** ☹: you need to keep all these CSPs around (vs. simpler/smaller states of DFS)

Searching by domain splitting

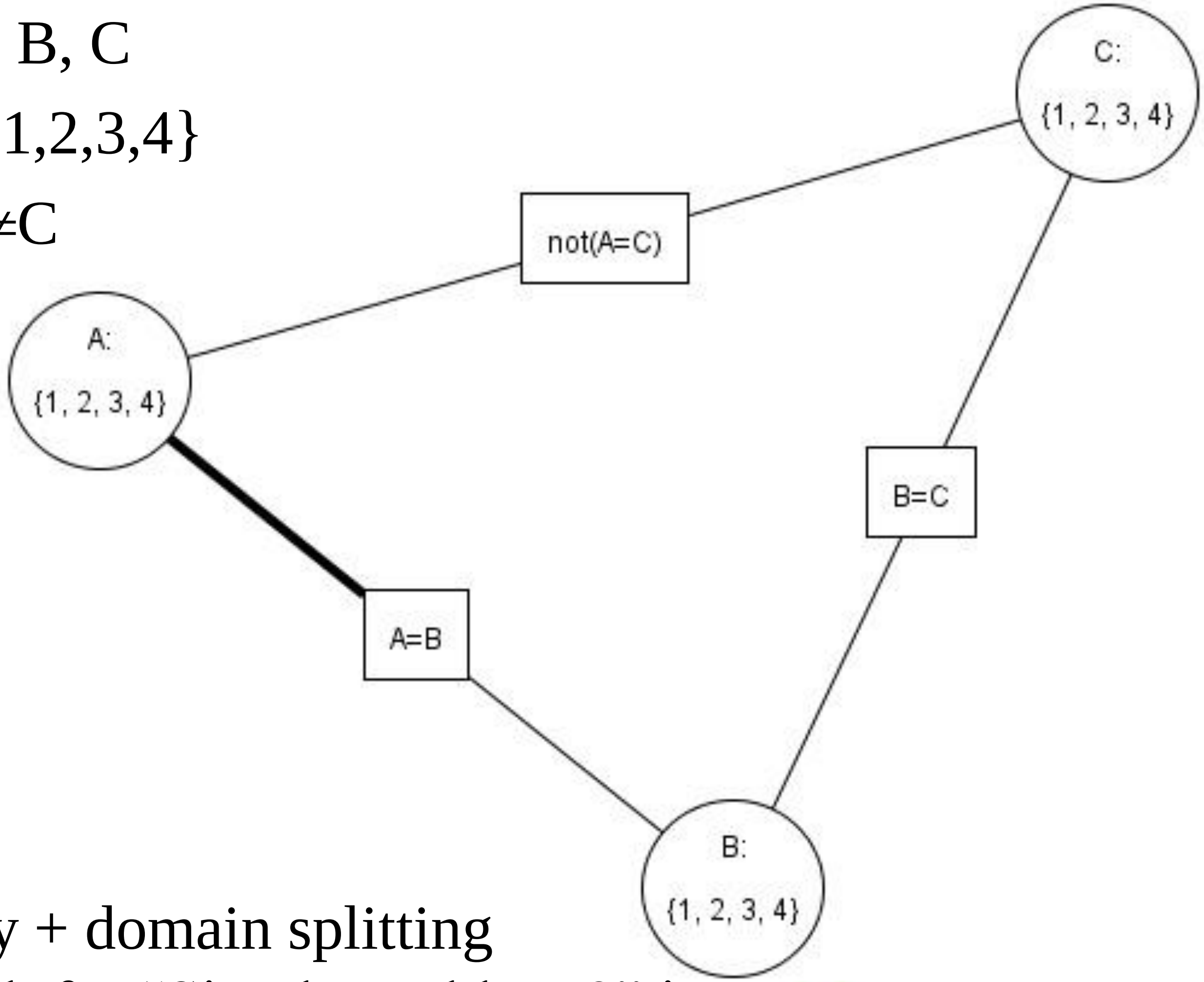


More formally: Arc consistency with domain splitting as another formulation of CSP as search

- **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**: any goal state

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

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

Next Class (Chpt. 4.7)

- **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

- **<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**