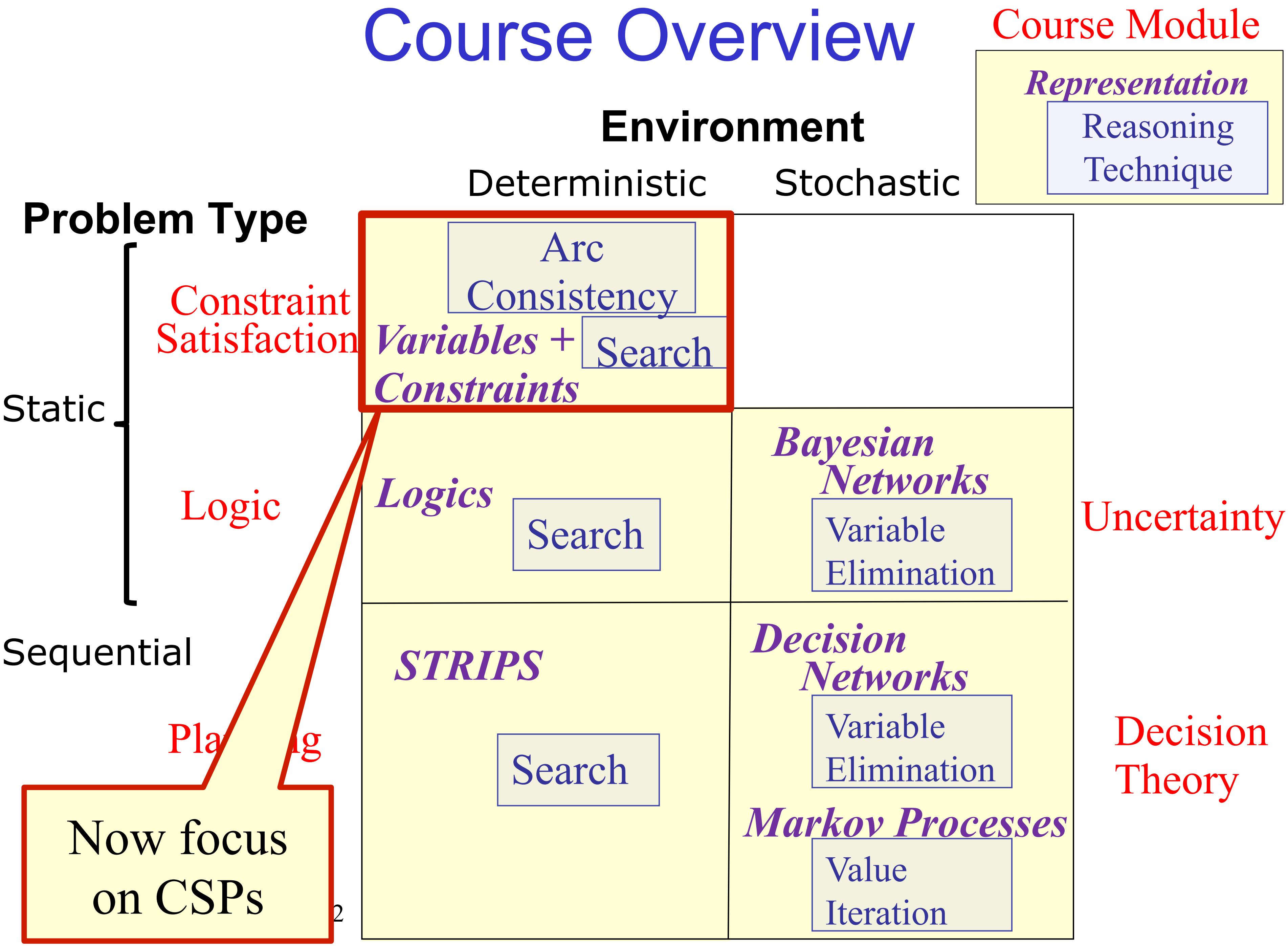
GAC Algorithm and Domain Splitting for CSPs

Alan Mackworth

UBC CS 322 – CSP 4
February 1, 2013

Textbook §4.6

Course Overview



Lecture Overview

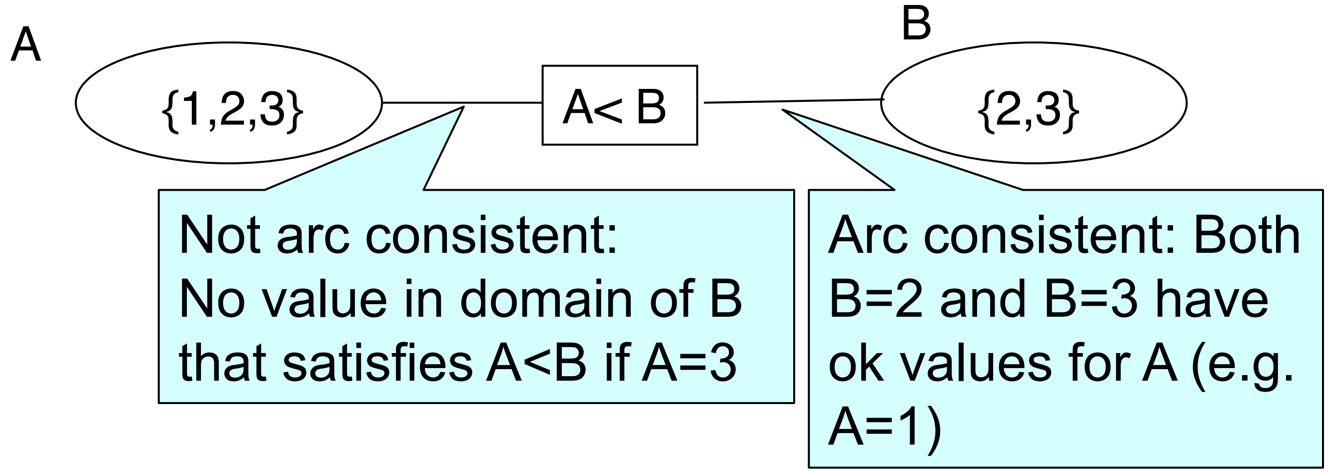
- Arc consistency
  - Recap
  - GAC algorithm
  - Complexity analysis
  - Domain splitting

Arc Consistency

Definitions:

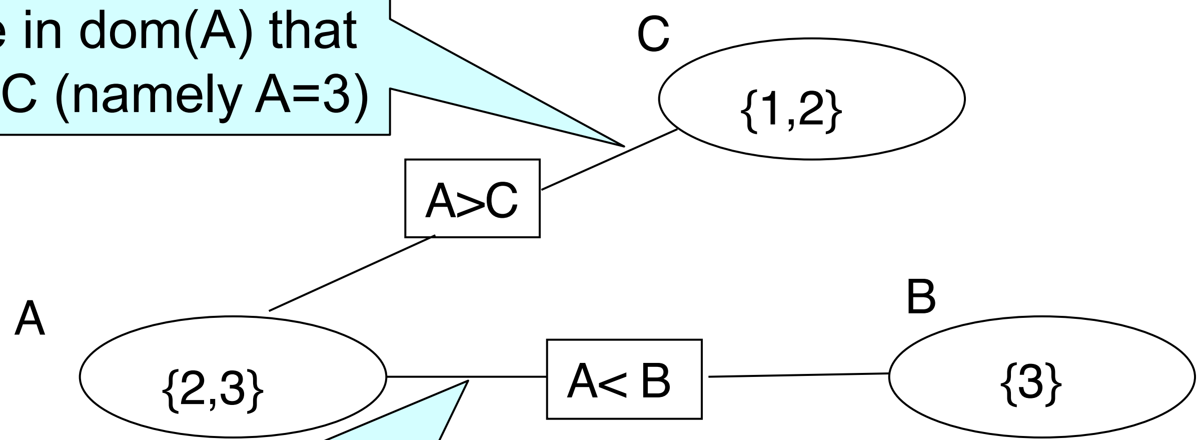
An arc <x, r(x,y)> is arc consistent if for each value x in dom(X) there is some value y in dom(Y) such that r(x,y) is satisfied.

A network is arc consistent if all its arcs are arc consistent.



Arc Consistency

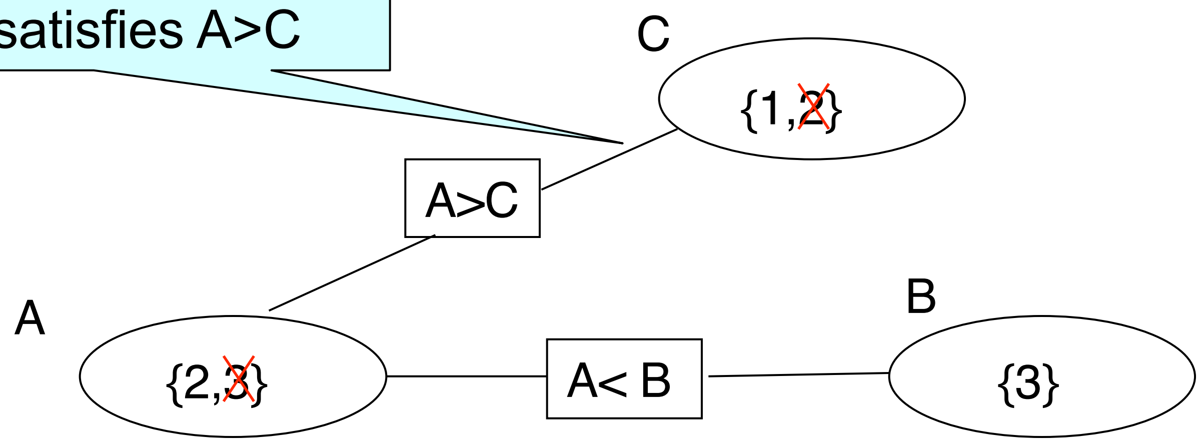
Arc consistent:
For each value in dom(C),
there is one in dom(A) that
satisfies A>C (namely A=3)



Not arc consistent:
No value in domain of B
that satisfies A<B if A=3

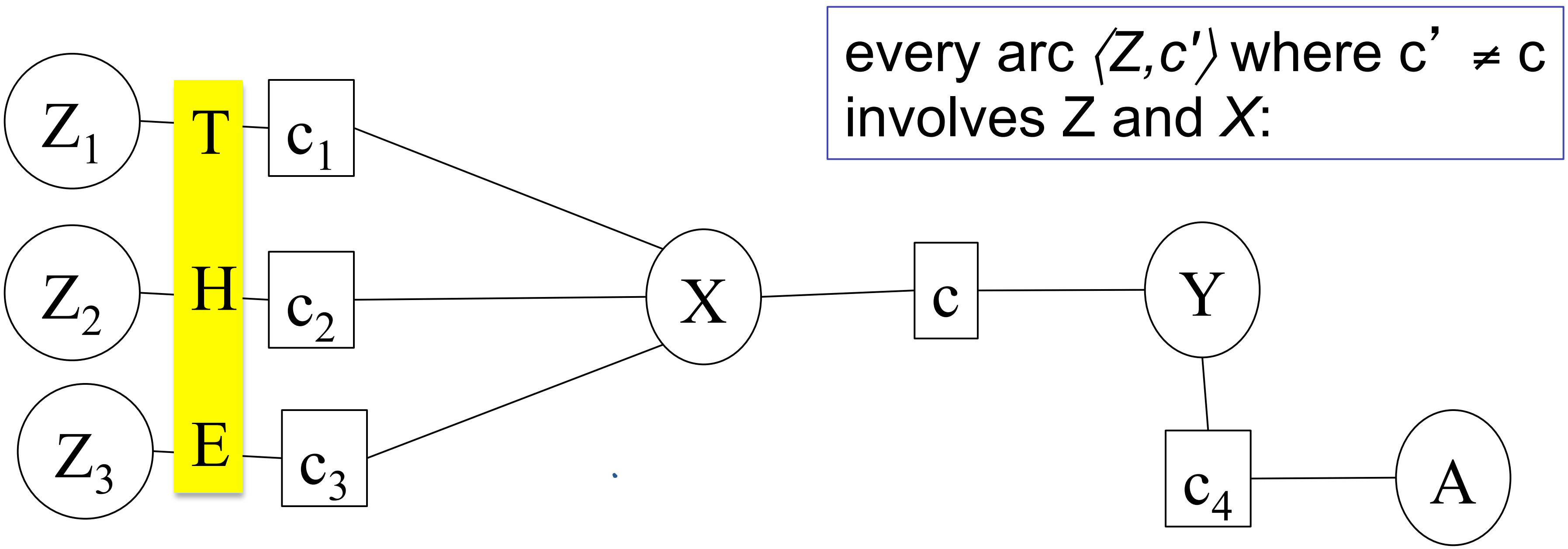
Arc Consistency

Not arc consistent anymore:
For C=2, there is no value in dom(A) that satisfies A>C



Which arcs need to be reconsidered?

- When we reduce the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, which arcs do we need to reconsider?



- You do not need to reconsider other arcs
  - If arc ⟨Y,c⟩ was arc consistent before, it will still be arc consistent
  - If an arc ⟨X,c'⟩ was arc consistent before, it will still be arc consistent
  - Nothing changes for arcs of constraints not involving X

Lecture Overview

- Arc consistency
  - Recap
  - GAC algorithm
  - Complexity analysis
  - Domain splitting

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
        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 is a value for y supporting x

X' s domain changed: ⇒ arcs (Z,c') for variables Z sharing a constraint c' with X could become inconsistent

Arc Consistency Algorithm: Interpreting Outcomes

Three possible outcomes (when all arcs are arc consistent):

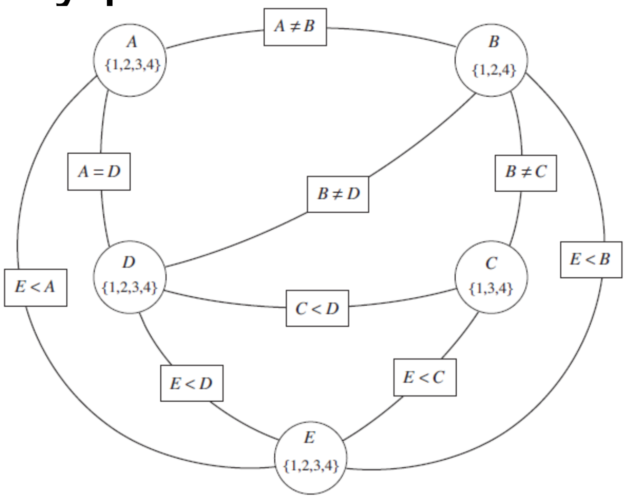
1. Each domain has a single value, e.g.
   http://www.cs.ubc.ca/~mack/CS322/AIspace/simple-network.xml
   (Download the file and load it as a local file in AIspace consistency applet)
   And "Scheduling Problem 1" in AIspace.
   We have a (unique) solution.

2. At least one domain is empty, e.g.
   http://www.cs.ubc.ca/~mack/CS322/AIspace/simple-infeasible.xml
   (All values are ruled out for this variable.)
   No solution!

3. Some domains have more than one value, e.g.
   built-in example "Simple Problem 2" or "Scheduling Problem 2"
   There may be one solution, many solutions, or *none*.
   Need to solve this new CSP (usually simpler) problem:
   same constraints, domains have been reduced

Arc Consistency Algorithm: Complexity

- Worst-case complexity of arc consistency procedure on a problem with N variables
  - let ***d*** be the max size of a variable domain
  - let ***c*** be the number of constraints



  - How often will we prune the domain of variable V? O(d) times
  - How many arcs will be put on the ToDoArc (TDA) list when pruning domain of variable V?
    - O(degree of variable V)
    - In total, across all variables: sum of degrees of all variables = … 2*number of constraints, i.e. 2*c
  - Together: we will only put O(dc) arcs on the ToDoArc list (2c arcs originally on TDA)
  - Checking consistency is $O(d^2)$ for each of them
- Overall complexity: $O(cd^3)$
- Compare to $O(d^N)$ of DFS! Arc consistency is MUCH faster.

Lecture Overview

- Arc consistency
  - Recap
  - GAC algorithm
  - Complexity analysis
  - Domain splitting

Can we have an arc consistent network with non-empty domains that has no solution?

YES

NO

- Example: vars A, B, C with domain {1, 2} and constraints $A \neq B$, $B \neq C$, $A \neq C$
- Or see AIspace CSP applet Simple Problem 2

Domain splitting (or case analysis)

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

     CSP with dom(X) = $\{x_1, x_2, x_3, x_4\}$ becomes

     CSP_1 with dom(X) = $\{x_1, x_2\}$ and
     CSP_2 with dom(X) = $\{x_3, x_4\}$

- Solution to CSP is the union of solutions to CSP_i

Whiteboard example for domain splitting

- …

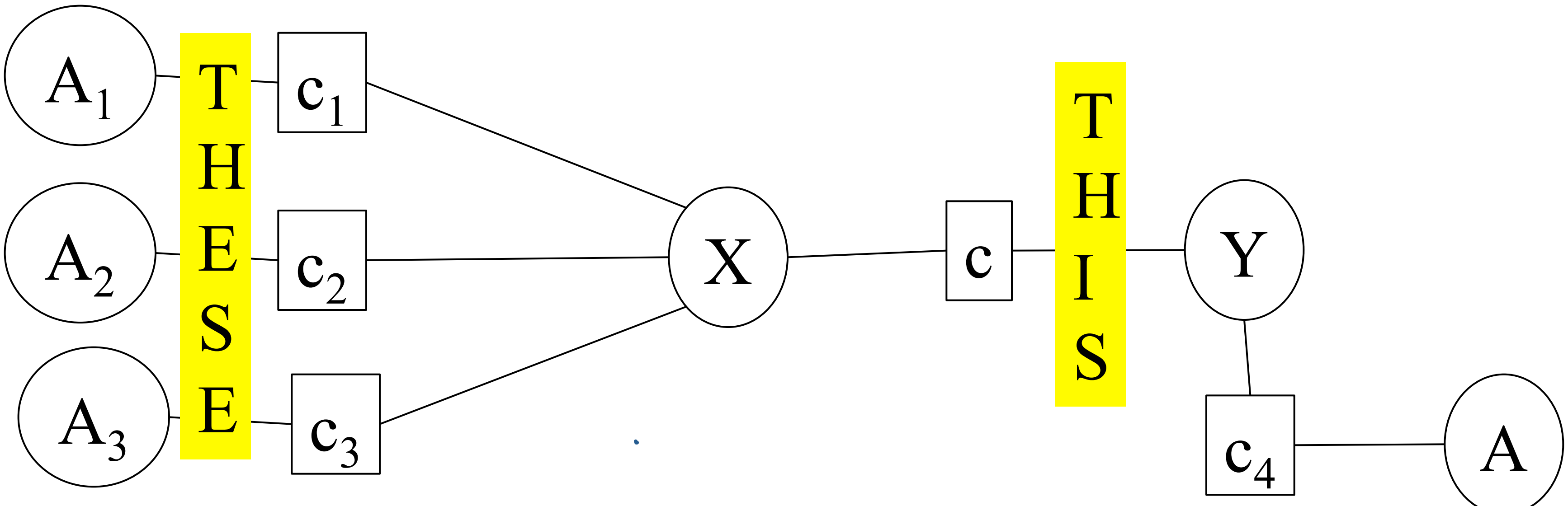
Domain splitting

- Each smaller CSP is easier to solve
  - Arc consistency might already solve it
- For each subCSP, which arcs have to be on the ToDoArcs list when we get the subCSP by splitting the domain of X?

arcs <Z, r(Z,X)>

arcs <Z, r(Z,X)> and <X, r(Z,X)>

All arcs



Domain splitting in action

- Trace it on "simple problem 2"

Searching by domain splitting

CSP, apply AC

If domains with multiple values

Split on one

CSP_1, apply AC

If domains with multiple values
Split on one

CSP_2, apply AC

If domains with multiple values…..Split on one

How many CSPs do we need to keep around at a time?
With depth m and 2 children at each split: O(2m). It's a DFS.

Learning Goals for today's class

- 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

---

- Coming up: local search, Section 4.8