Lecture 9

Arc Consistency

(4.5, 4.6)

Lecture Overview

- Recap of Lecture 8
- Arc Consistency for CSP
- Domain Splitting

Course Overview

Representation

Environment

Deterministic Stochastic

ReasoningTechnique

Problem Type $\boxed{\text{Arc}}$

Consistency

We'll start

CPS

Constraint Satisfaction *Vars Constraints* + Search from

Static ***Belief Nets***

Query ***Logics*** Variable

Search

Elimination

Sequential ***STRIPS*** ***Decision Nets*** Variable

Planning Elimination

Search

Markov Processes

Value

Iteration

- Constraint Satisfaction Problems (CPS):
  - State
  - Successor function
  - Goal test
  - Solution
  - Heuristic function

We will look at Search for CSP

- Query :
- State
- Successor function
- Goal test
- Solution
- Heuristic function
- Planning

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

Constraint Satisfaction Problems (CSPs): Definitions

Definition:

A constraint satisfaction problem (CSP) consists of:

- a set of variables V
- a domain dom(V) for each variable
- a set of constraints C

- Constraints are restrictions on the values that one or more variables can take
- Unary constraint: restriction involving a single variable

- k-ary constraint: restriction involving k different variables
  - ✓We will mostly deal with binary constraints
- Constraints can be specified by
  1. listing all combinations of valid domain values for the variables participating in the constraint
  2. giving a function that returns true when given values for each variable which satisfy the constraint

Example: Map-Coloring

Variables WA, NT, Q, NSW, V, SA, T

Domains D_i= {red,green,blue}

Constraints: adjacent regions must have different colors

e.g., WA ≠ NT, NT ≠ SA, NT ≠ QU, ……,

Or

WA	NT
Red	Green
Red	Bue
Green	Red
Green	Blue
Blue	Red
Blue	Green

NT	SA
Red	Green
Red	Bue
Green	Red
Green	Blue
Blue	Red
Blue	Green

NT	QU
Red	Green
Red	Bue
Green	Red
Green	Blue
Blue	Red
Blue	Green

…………….

Constraint Satisfaction Problems (CSPs): Definitions

Definition:

A constraint satisfaction problem (CSP) consists of:

- a set of variables V
- a domain dom(V) for each variable
- a set of constraints C

Definition:

A model of a CSP is an assignment of values to all of its variables (i.e., a possible world) that satisfies all of its constraints.

WA = red,

NT = green,

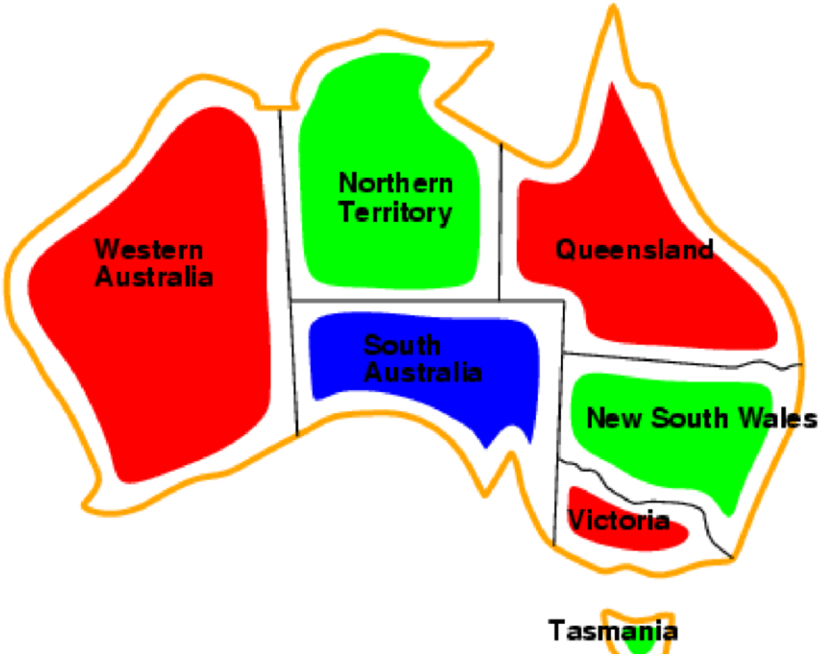
Q = red,

NSW = green,

V = red,

SA = blue,

T = green



Solving Constraint Satisfaction Problems

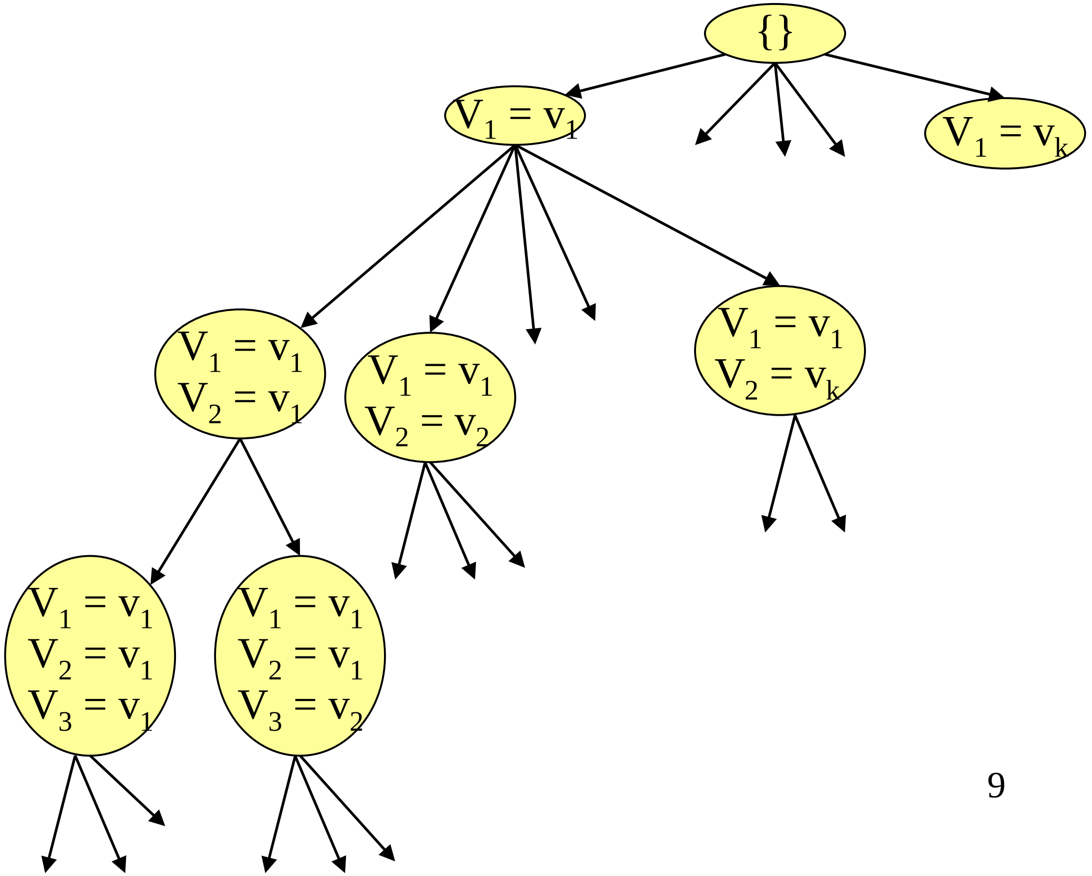
- Even the simplest problem of determining whether or not a model exists in a general CSP with finite domains is NPhard
- There is no known algorithm with worst case polynomial runtime.

- However, we can try to:
- identify special cases for which algorithms are efficient
- find efficient (polynomial) consistency algorithms that reduce the size of the search space
- work on approximation algorithms that can find good solutions quickly, even though they may offer no theoretical guarantees
- find algorithms that are fast on typical (not worst case) cases

Search-Based Approach

- Constraint Satisfaction (Problems):
- State: assignments of values to a subset of the variables
- Successor function: assign values to a “free” variable
- Goal test: all variables assigned a value and all constraints satisfied?
- Solution: possible world that satisfies the constraints
- Heuristic function: none (all solutions at the same distance from start)

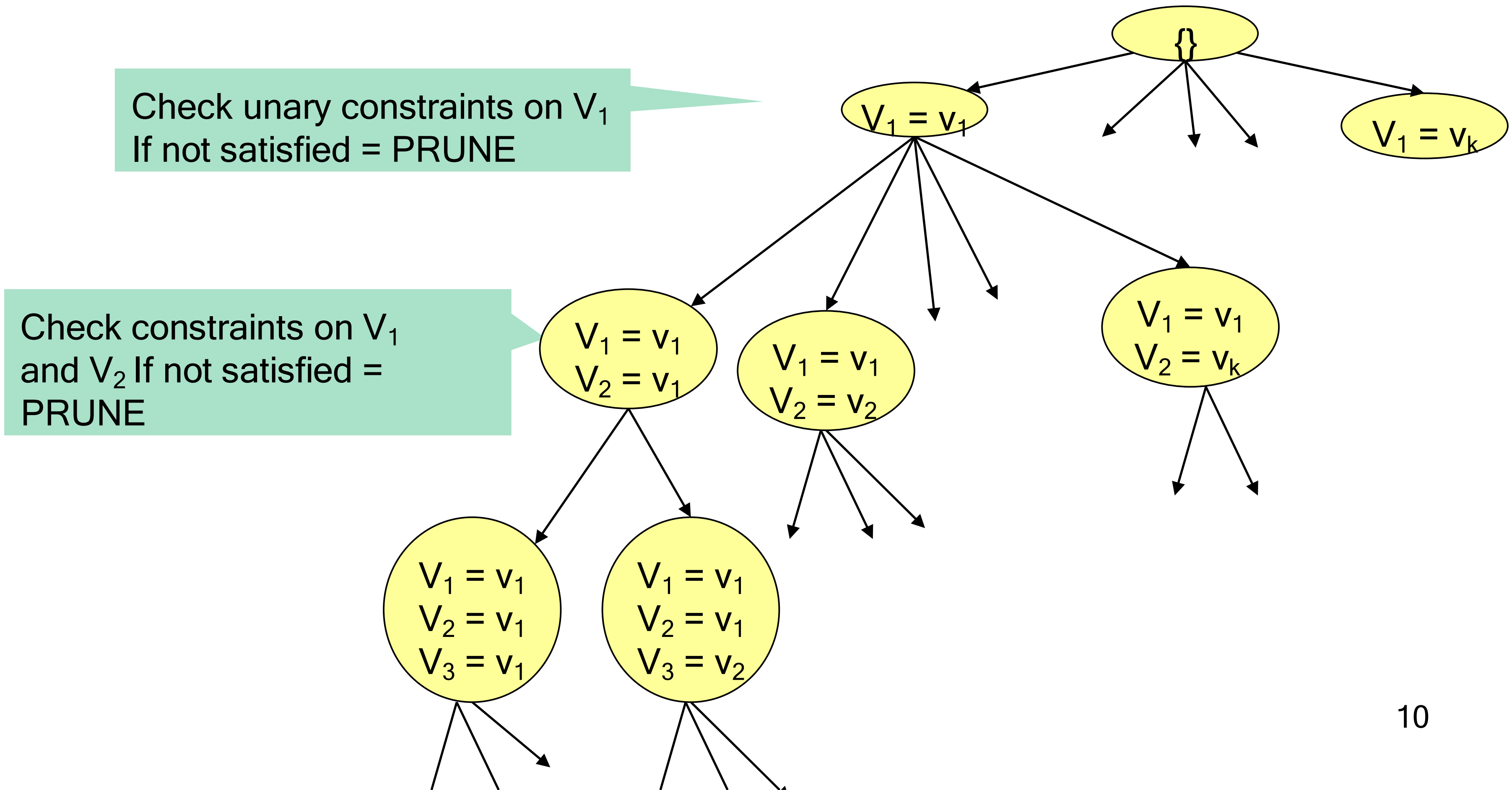
- Planning :
- State
- Successor function
- Goal test
- Solution
- Heuristic function
- Inference
- State
- Successor function
- Goal test
- Solution
- Heuristic function



9

Backtracking algorithms

- Explore search space via DFS but evaluate each constraint as soon as all its variables are bound.



- Any partial assignment that doesn't satisfy the constraint can be pruned.

Selecting variables in a smart way

- Backtracking relies on one or more heuristics to select which variables to consider next. - E.g, variable involved in the largest number of constraints:

  “If you are going to fail on this branch, fail early!”

- But we will look at an alternative approach that can do much better

- Arc Consistency:
- Key idea: prune the domains as much as possible before searching for a solution.

Lecture Overview

- Recap of Lecture 8
- Arc Consistency for CSP
- Domain Splitting

Can we do better than Search?

Key idea

- prune the domains as much as possible before searching for a solution.

Definition: A variable is domain consistent if no value of its domain is ruled impossible by any unary constraints.

- Example: dom(V) = {1, 2, 3, 4}.
- Variable V is not domain consistent with the constraint $V \neq 2$

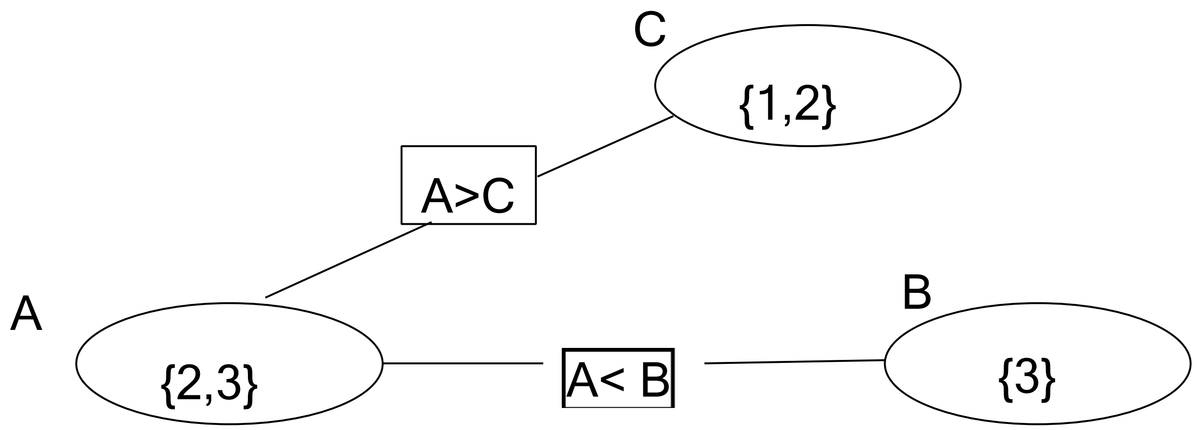
- It is domain consistent once we remove 2 from its domain.

Pruning domains is trivial for unary constraints. Trickier for k-ary ones.

Constraint Networks

Def. A constraint network is defined by a graph, with

- one node for every variable (drawn as circle)
- one node for every constraint (drawn as rectangle)
- undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.



- Three variables: A, B, C
- Two constraints: A<B,
  A>C

Example Constraint Network

Def. A constraint network is defined by a graph, with

- one node for every variable (drawn as circle)
- one node for every constraint (drawn as rectangle)
- undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

Example:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- 3 Constraints: $A < B$, $B < C$, $B = 3$ 5 edges/arcs in the constraint

network:

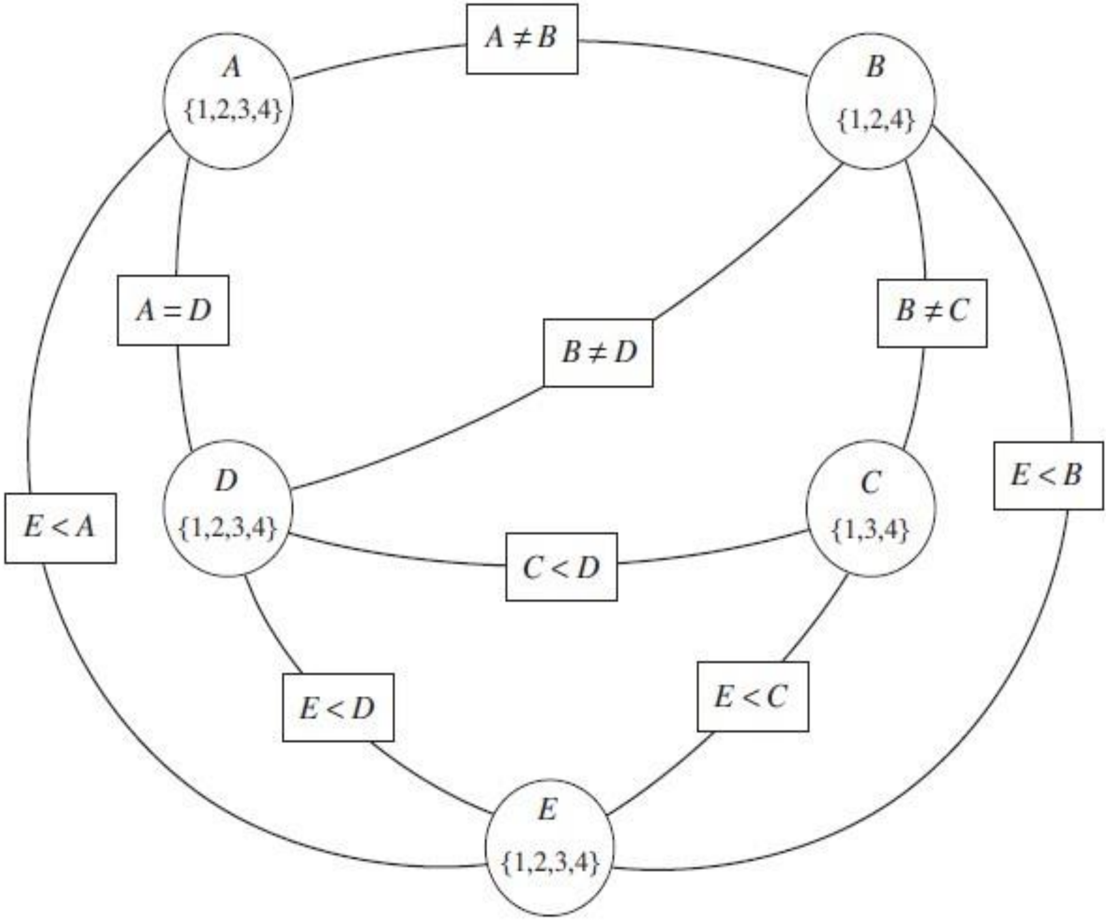
$\langle A,A<B\rangle$, $\langle B,A<B\rangle$

$\langle B,B<C\rangle$, $\langle C,B<C\rangle$

$\langle B, B=3\rangle$

A more complicated example

How many variables are there in this constraint network?



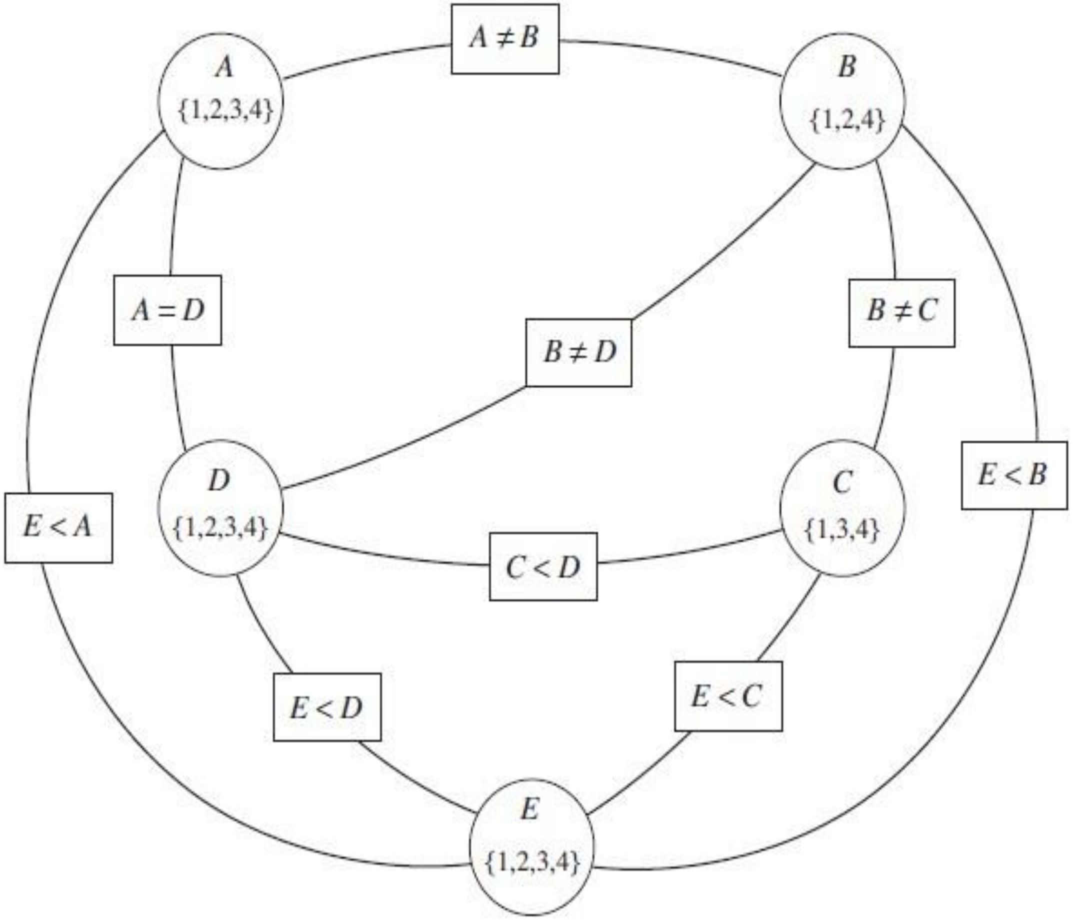
A more complicated example

How many variables are there in this constraint network?

A. 5 B. 6

C. 9 D. 14

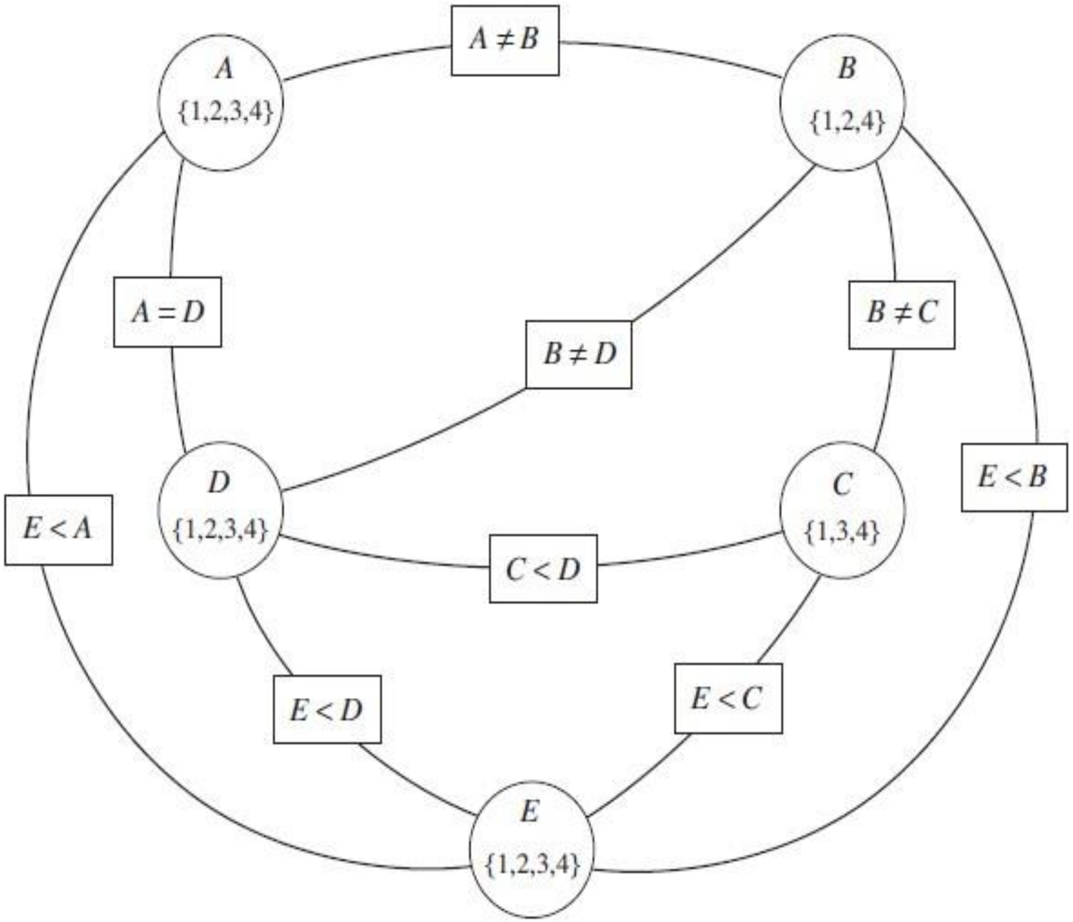
How many variables are there in this constraint network?



A more complicated example

A. 5 B.6

C. 9 D. 14



A more complicated example

D. 18

A more complicated example

How many variables are there in this constraint network?

A. 5

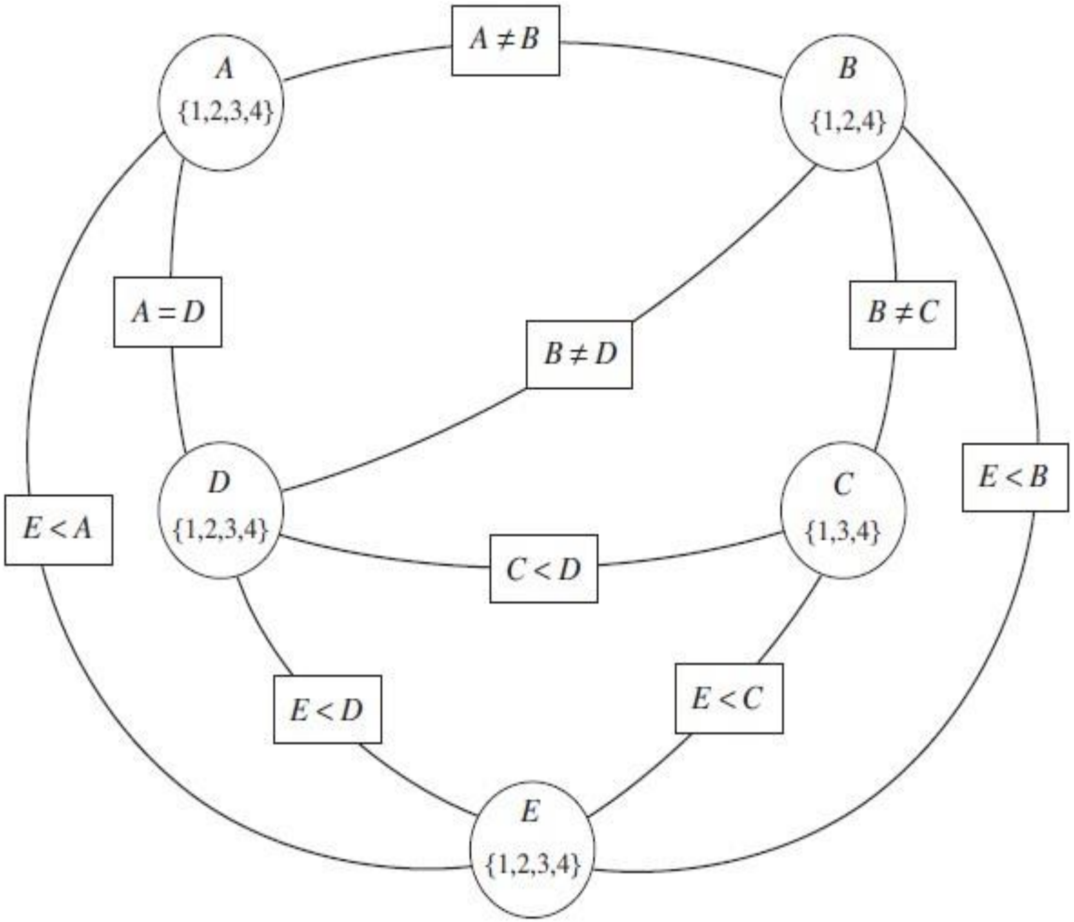
A. 6

B. 9

C. 14

A more complicated example

How many constraints?

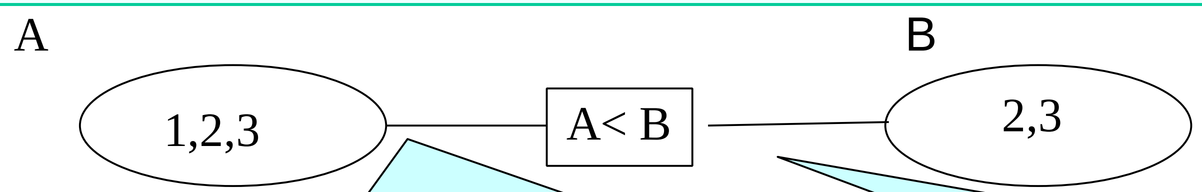


Arc Consistency

Definition:

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.



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

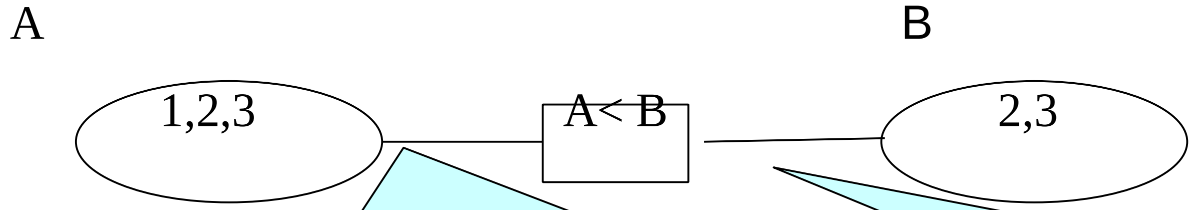
Arc consistent: Both B=2 and B=3 have ok values for A
- e.g. for A = 1

Arc Consistency

Definition:

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.



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

A = 3

Arc consistent: Both B=2 and B=3 have ok values for A

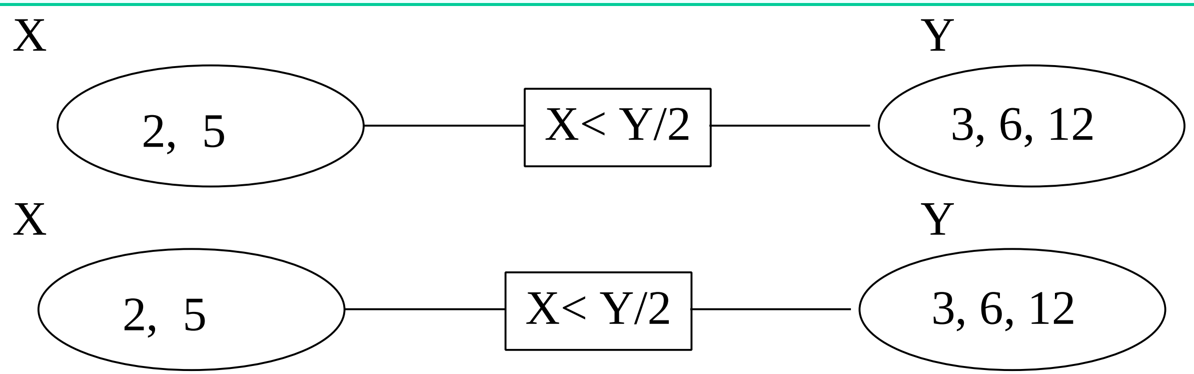
- e.g. for B=2 A = 1

Arc Consistency

Definition:

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.



A. Both

arcs are consistent

Arc Consistency

Definition:

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.

B. Left consistent, right inconsistent

C. Left inconsistent, right consistent

D. Both arcs are inconsistent

Left = <X, (X < Y/2)>

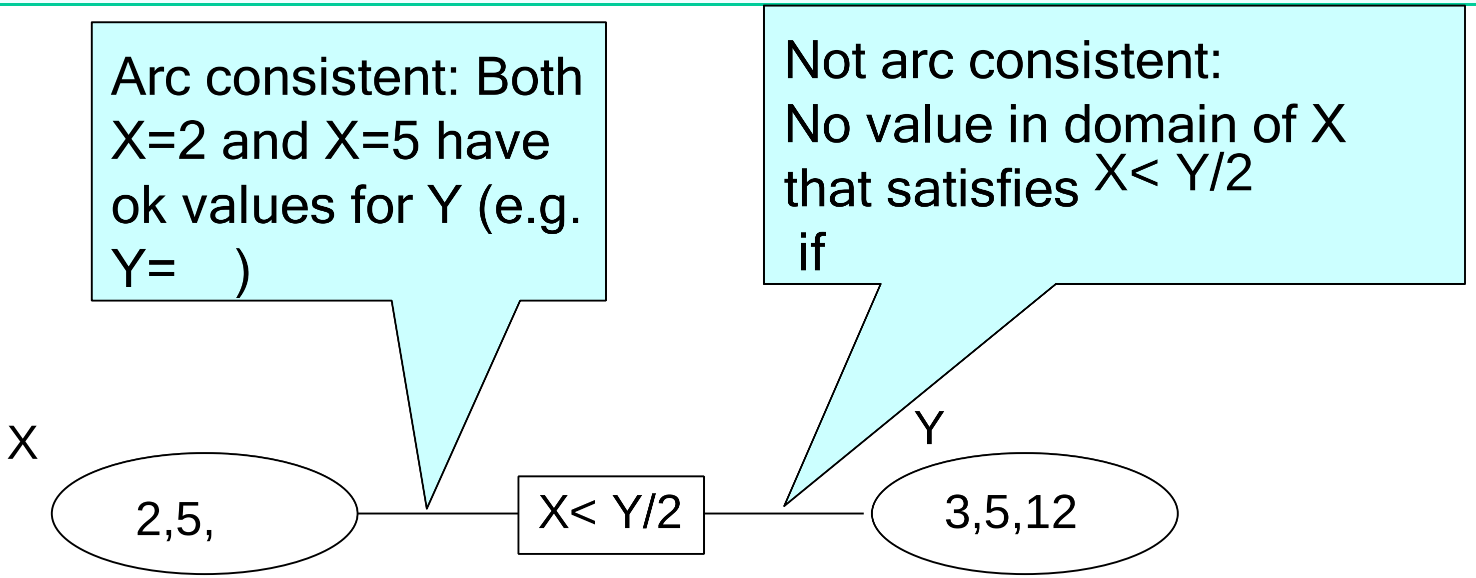
Right = <Y, (X < Y/2)>

Arc Consistency

Definition:

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.



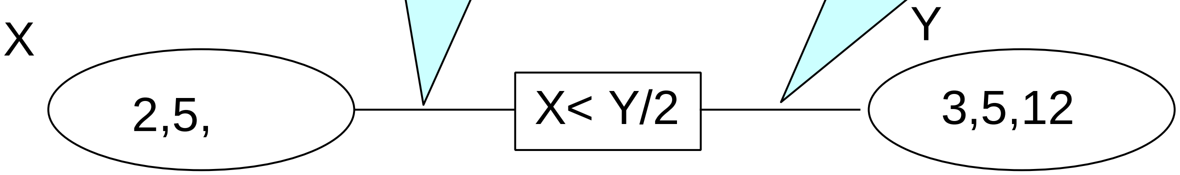
Definition:

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 consistent: Both X=2 and X=5 have ok values for Y (e.g. Y= 12)

Not arc consistent:
No value in domain of X that satisfies X< Y/2 if Y = 3



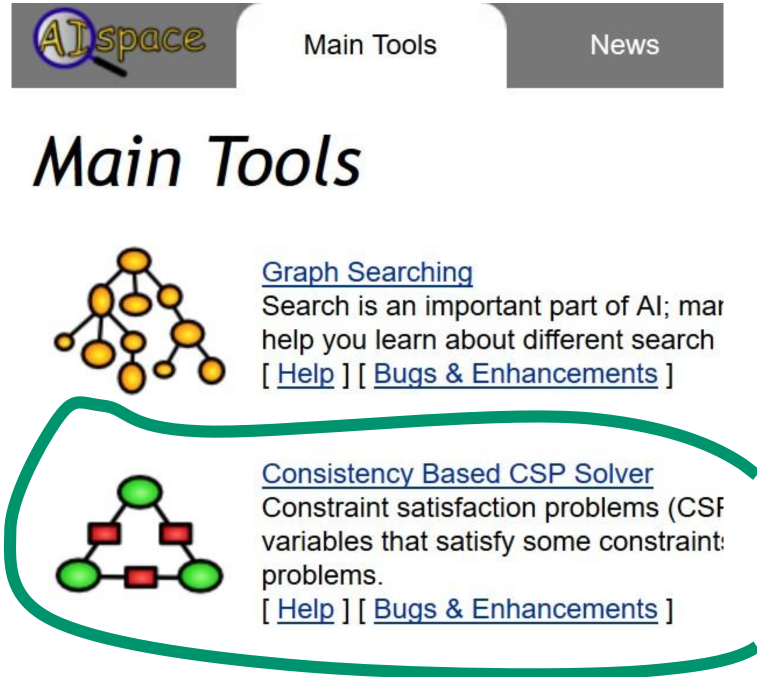
Arc Consistency
Arc Consistency Algorithm

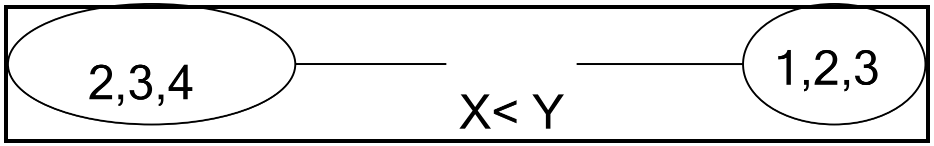
How can we enforce Arc Consistency?

- If an arc <X, r(X,Y)>is not arc consistent
  - Delete all values xin dom(X)for which there is no corresponding value in dom(Y)
  - This deletion makes the arc <X, r(X,Y)> arc consistent.
  - This removal can never rule out any models/solutions

WHY?

XY





Download this example (SimpleCSP) from Schedule page

Save to a local file and then open file in the Aispace applet Consistency Based CSP Solver

Arc Consistency Algorithm

How can we enforce Arc Consistency?

- If an arc <X, r(X,Y)>is not arc consistent
  - Delete all values xin dom(X)for which there is no corresponding value in dom(Y)
  - This deletion makes the arc <X, r(X,Y)> arc consistent.
  - This removal can never rule out any models/solutions

Algorithm: general idea

- Go through all the arcs in the network

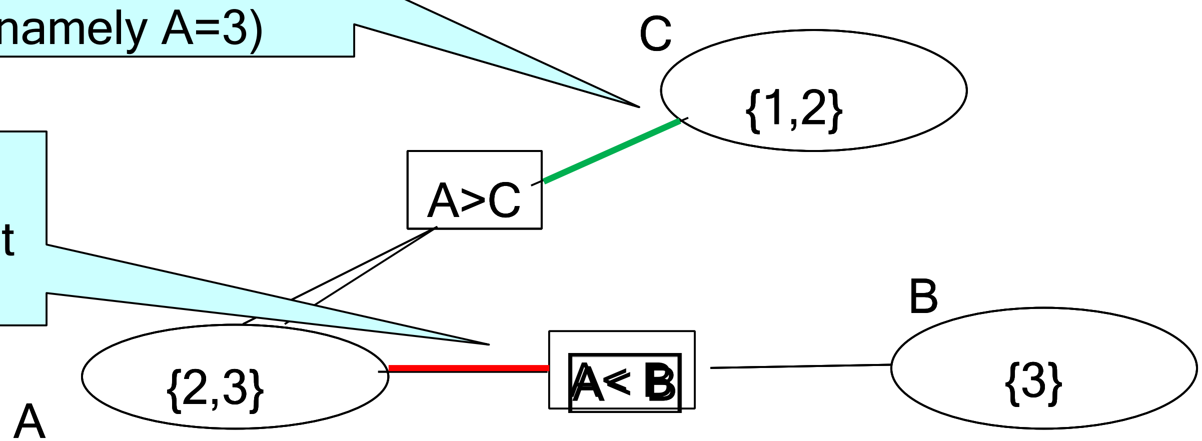
Arc Consistency

- Make each arc consistent by pruning the appropriate domain, when needed
- Reconsider consistent arcs that could be made inconsistent again by this pruning
- Eventually reach a 'fixed point': all arcs consistent 27

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

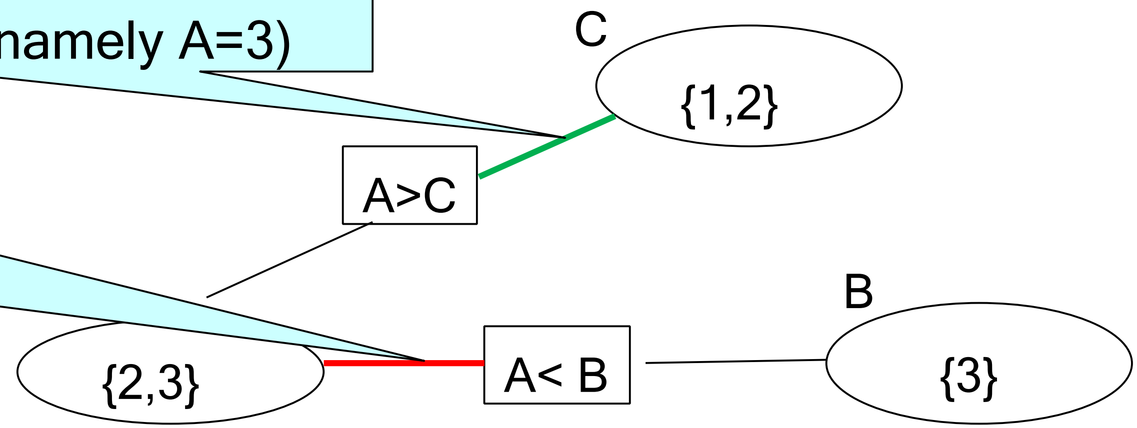


Try to build this simple network in AISpace Try to build this simple network in AISpace

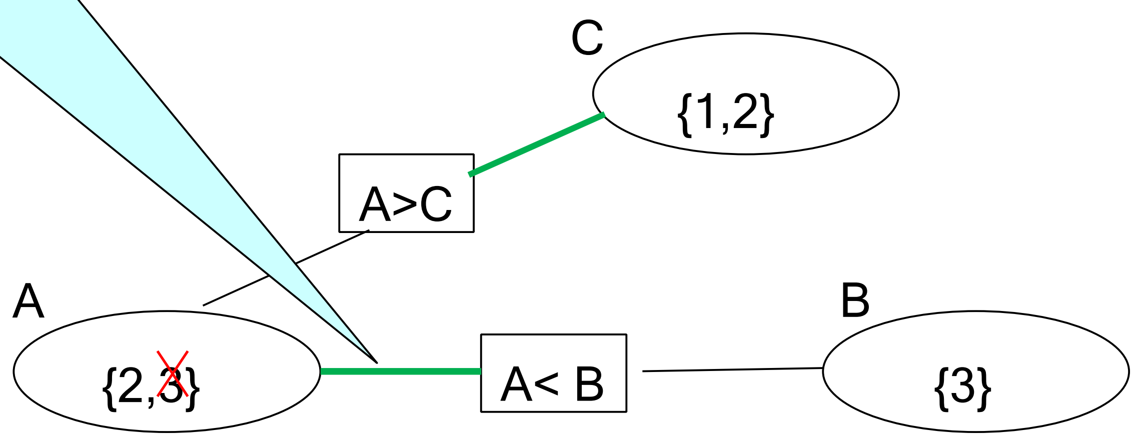
Arc Consistency

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



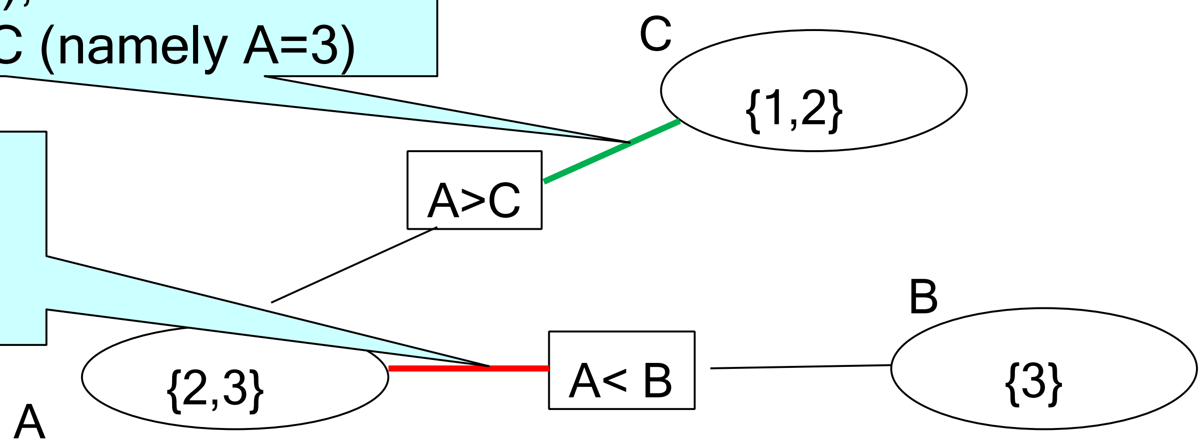
Pruning A=3 makes this arc consistent



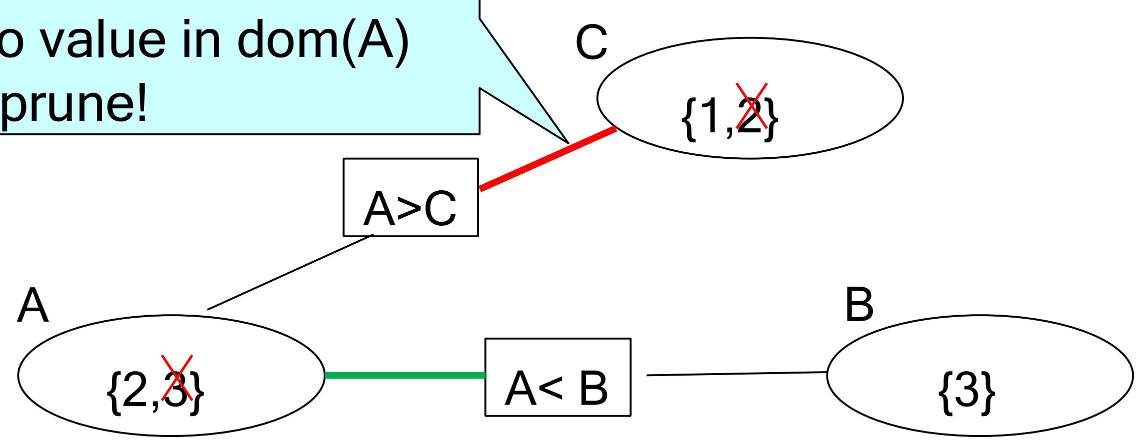
Which arcs need to be reconsidered?

- 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



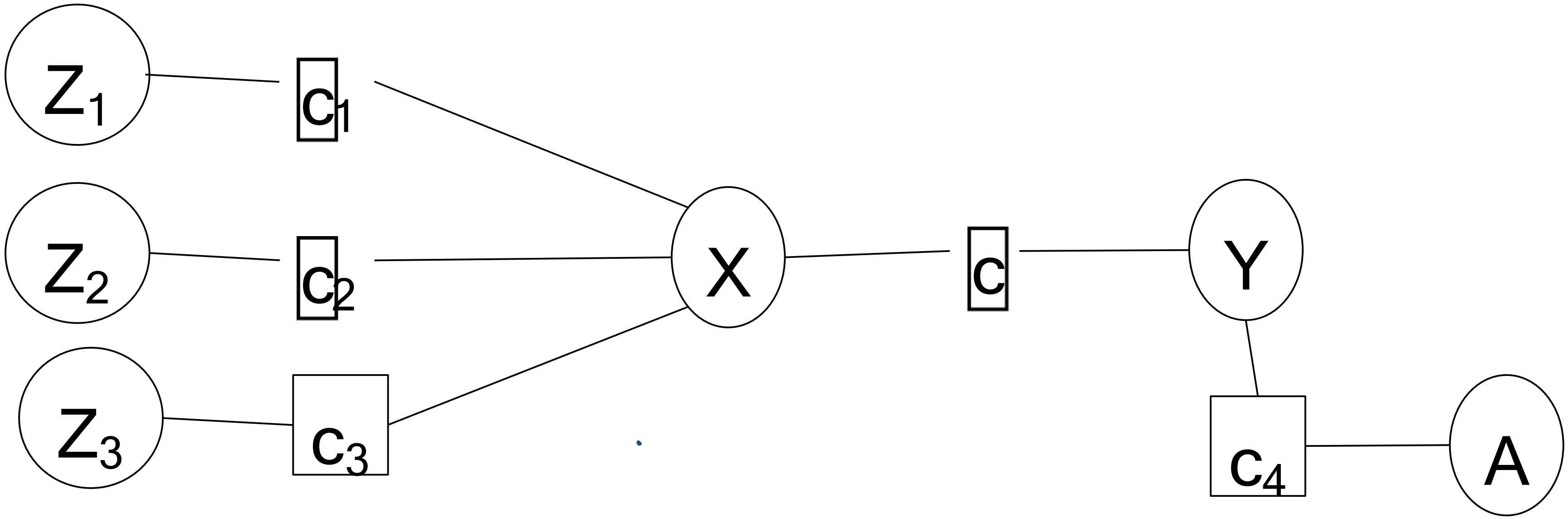
But after pruning A= 3: Not arc consistent anymore:
For C=2, there is no value in dom(A) that satisfies A>C: prune!



Arc Consistency

For the constraint network below, assume that

- Arc Consistency reduces the domain of variable X to make arc ⟨X,c⟩ arc consistent



Which arcs need to be reconsidered?

-
  - All other arcs in the figure were already consistent Which of these other arcs need to be reconsidered?

Which arcs need to be reconsidered?

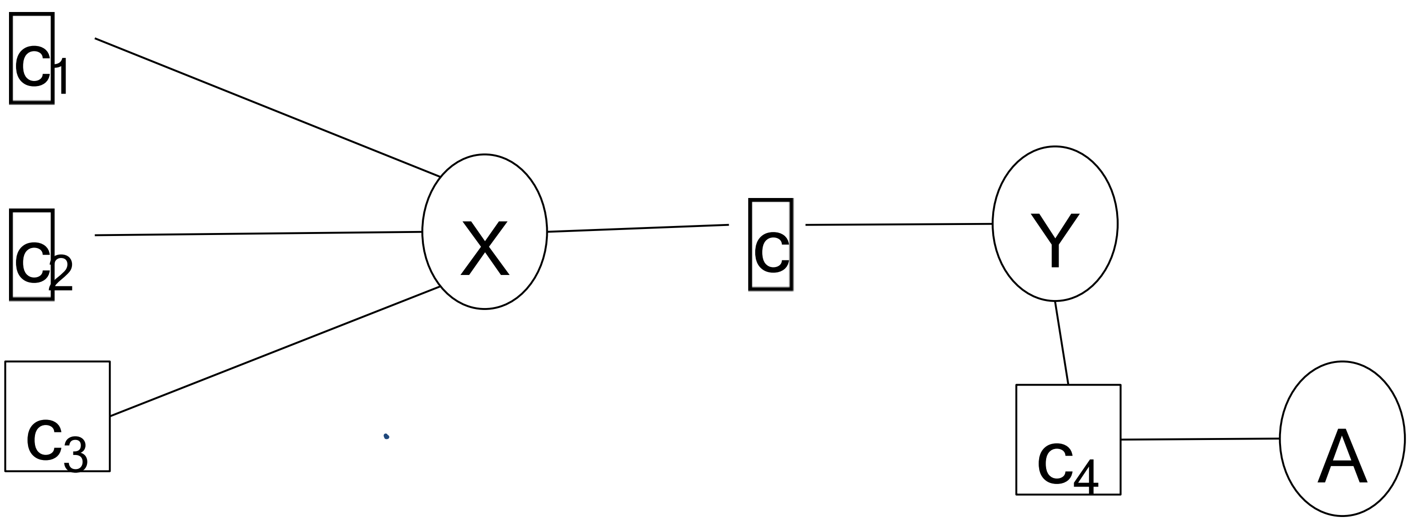
- When Arc Consistency reduces the domain of variable X to make arc ⟨X,c⟩ arc consistent, it need to reconsider the following arcs (that were already consistent)

every arc ⟨Z_i, c_i⟩where c
involves Z and X:

A. Yes all of them

B. None of them

C. Only some of them



Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, does it need to reconsider the following arcs (that were already consistent)?

Z_1

Z_2

Z_3

THESE?

D. It depends on the constraint c

A. Yes all of them

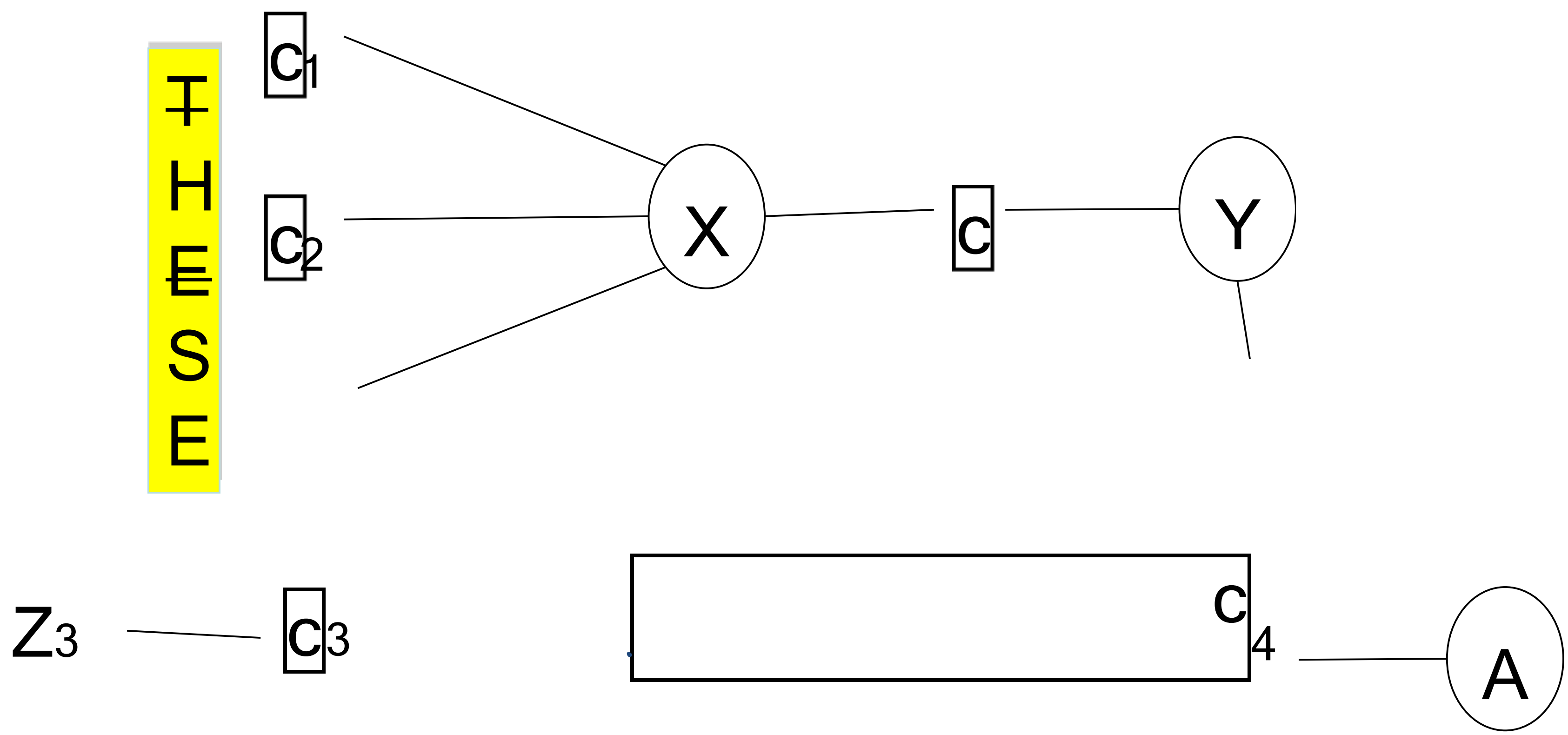
Z3 — C3

C4 — A

Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

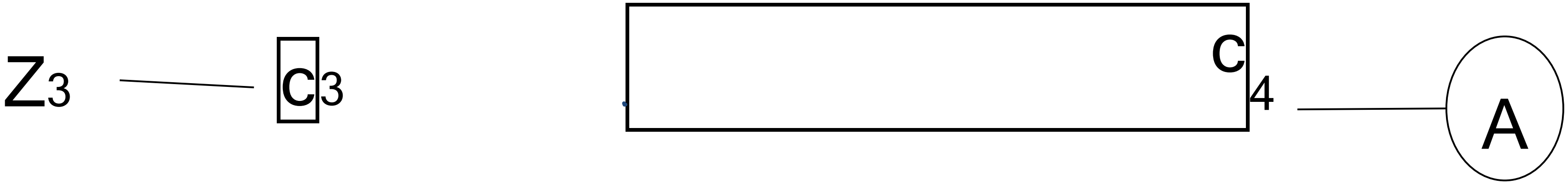
every arc $\langle Z_i, c_i \rangle$ where c' ≠ c
involves Z and X:



Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

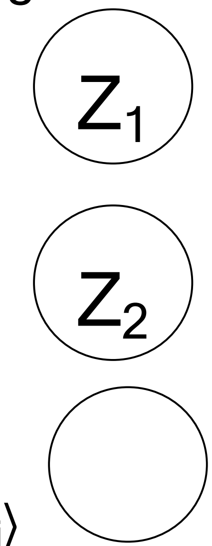
?



Which arcs need to be reconsidered?

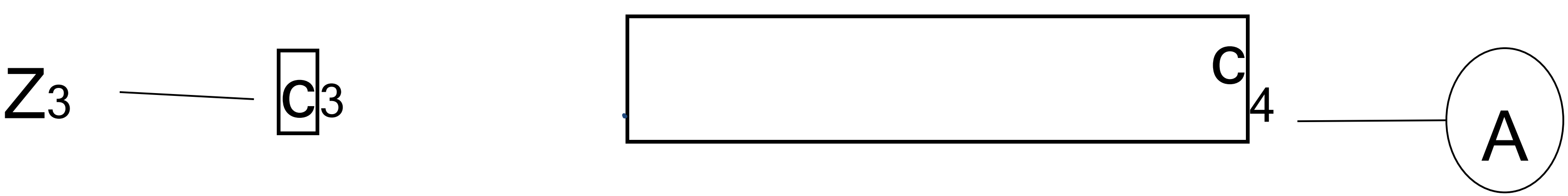
- When Arc Consistency reduces the domain of a variable X to make an

Pruning elements of Dom(x) may remove



values that made one or more of $\langle Z_i, c_i \rangle$ consistent

35



Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, does it need to reconsider the following arcs (that were already consistent)?

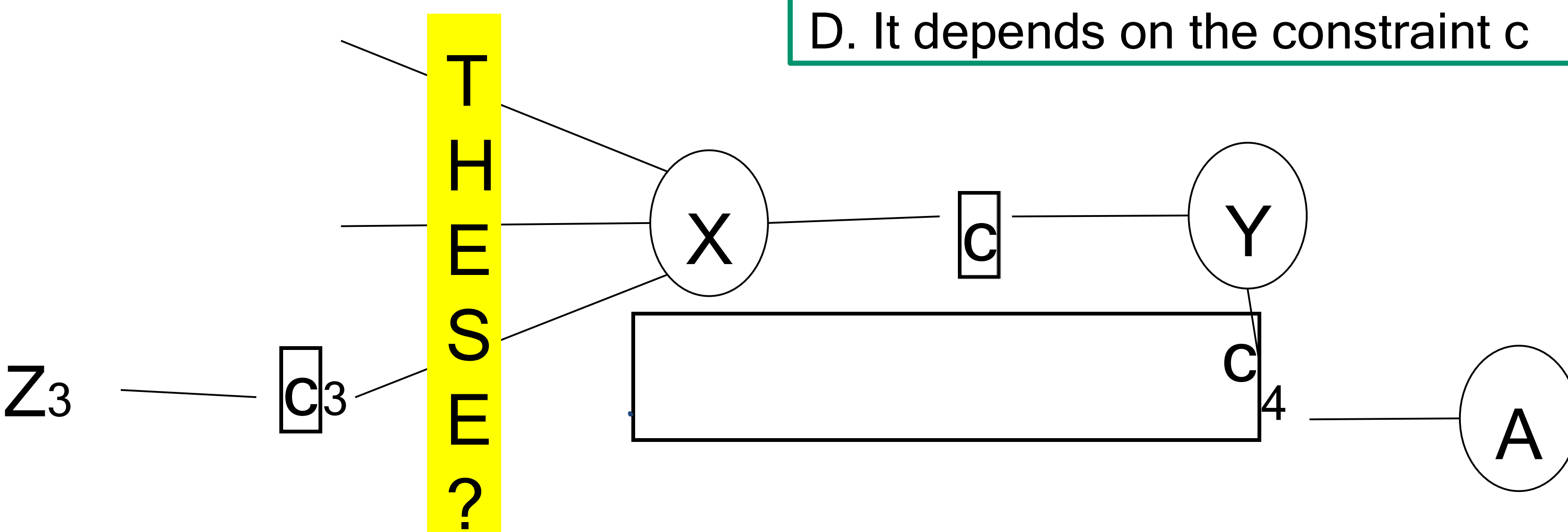
A. Yes all of them

every arc ⟨X, c_i⟩where c'

B. None of them

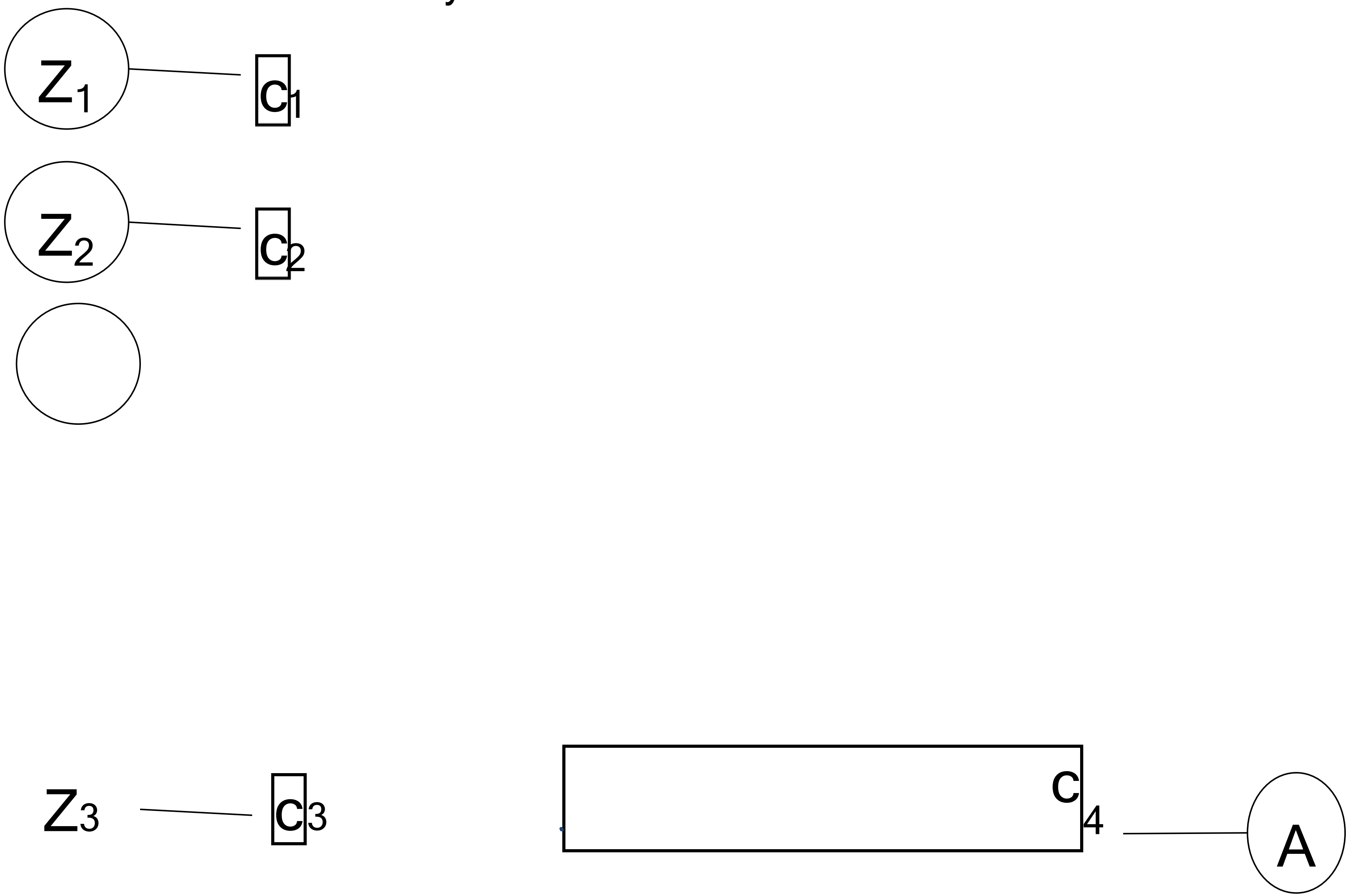
C. Only some of them

D. It depends on the constraint c



Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

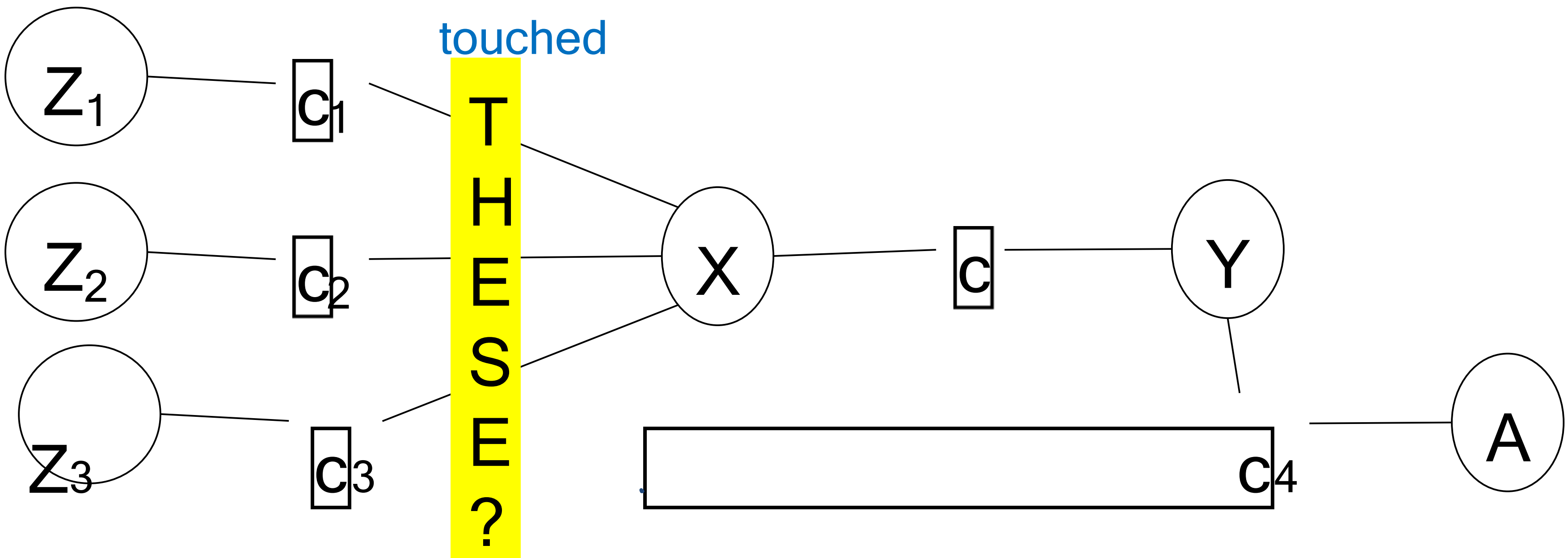


Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

  arc ⟨X,c⟩ arc consistent, does it need to reconsider the following arcs (that were already consistent)?

every arc $\langle X,c_i \rangle$ where c’ ≠ c



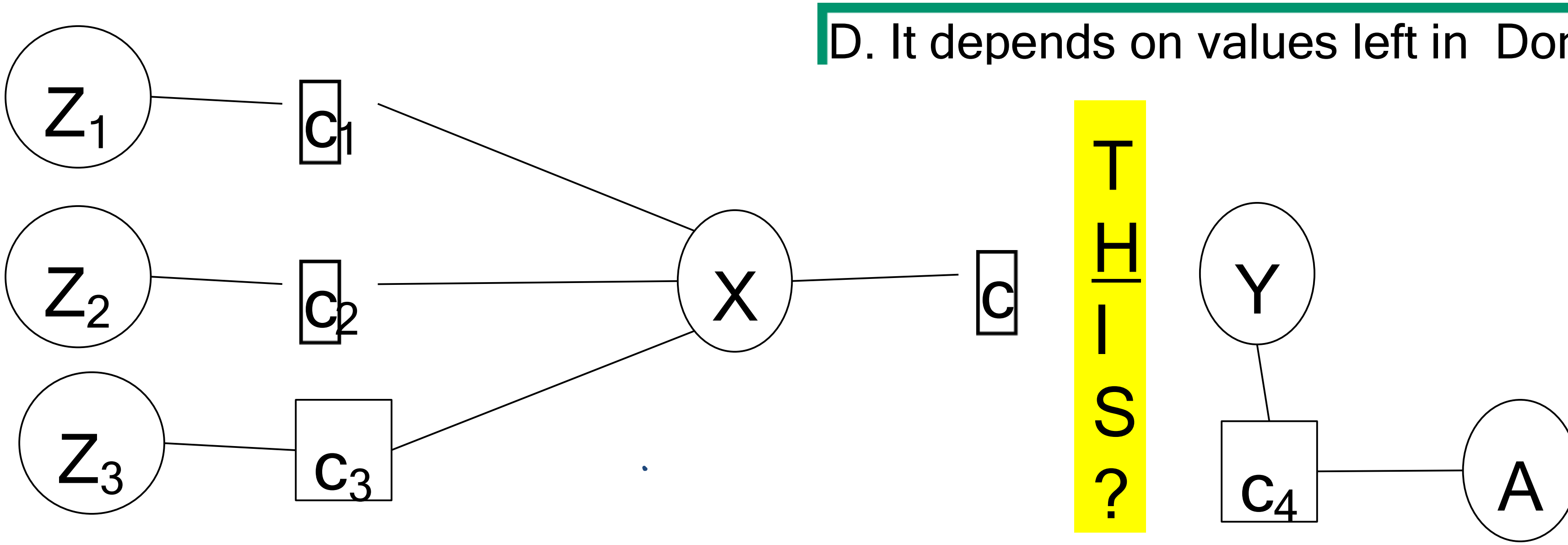
Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, does it need to reconsider the following arcs (that were already consistent)?

B. None of them

If an arc ⟨X,c_i⟩ was arc consistent before, it will still be arc consistent. The domains of Z_i have not been

D. It depends on values left in Dom(X)



Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

The arc ⟨Y, c⟩related to the constraint c involved in ⟨X, c⟩, which caused the pruning of Dom(X) :

A. Yes

B. No

C. It depends on the constraint c

Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an

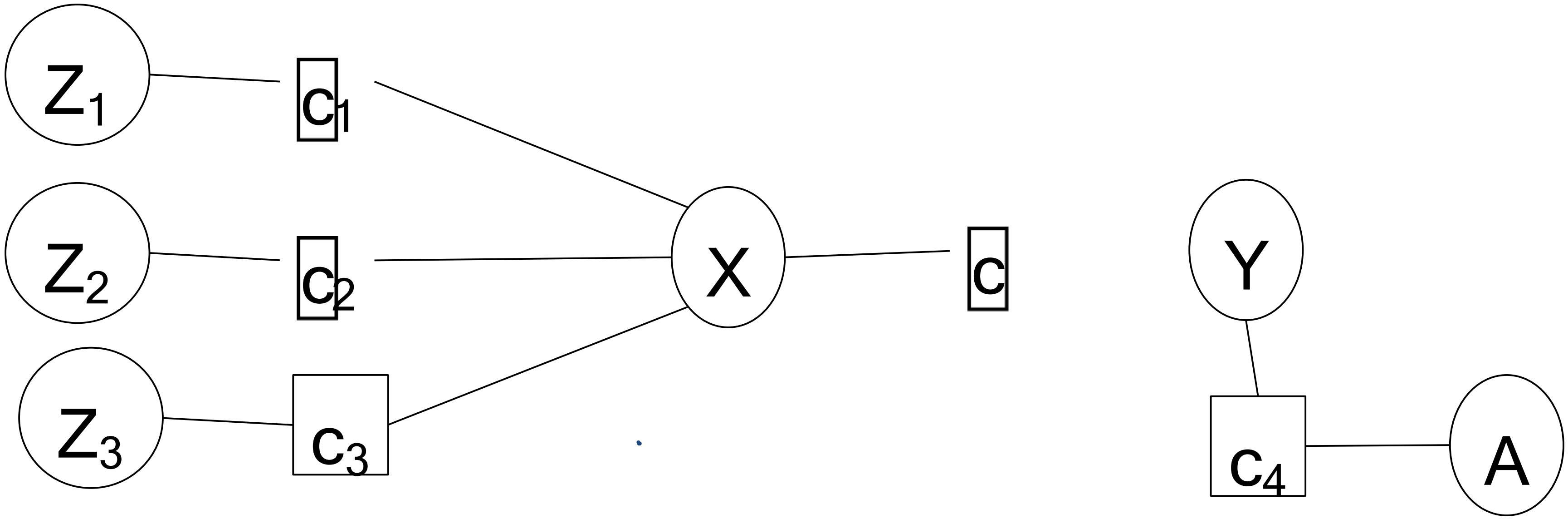
T
~~H~~
I

Which arcs need to be reconsidered?

- When Arc Consistency reduces the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, does it need to reconsider the following arcs (that were already consistent)?

S

B. No



Which arcs need to be reconsidered?

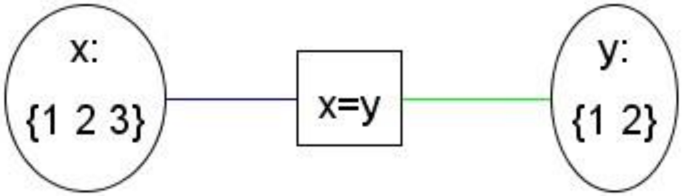
- When Arc Consistency reduces the domain of a variable X to make an

The arc ⟨Y, c⟩related to the constraint c involved in ⟨X, c⟩, which caused the pruning of Dom(X) :

If arc ⟨Y,c⟩ was arc consistent before, it will still be arc consistent

"Consistent before" means each element y_i in Y must have an element x_i in X that satisfies the constraint. Those x_i would not be pruned from Dom(X), so arc ⟨Y,c⟩ stays consistent

Specific Example

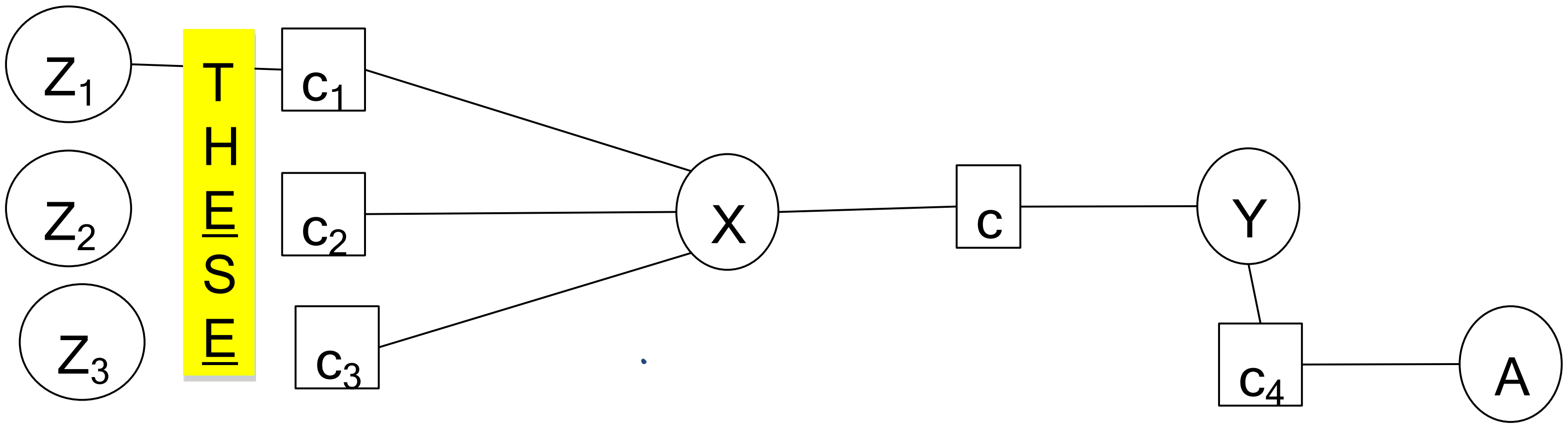


Arc <Y, x=y> is consistent because each value of Y (1,2) has a corresponding value in X that satisfies x =y

Arc <X, x=y> is not consistent so X needs to be pruned in relation to this arc (not some other arc in the network). This is the situation described in the last clicker question. But only items that were not involved in making <Y, x=y> consistent may end up being pruned (3 in this case). Those who were involved (i.e. 1,2 here) *must* have a counterpart value in Y, since <Y, x=y> is consistent

Which arcs need to be reconsidered?

- When AC reduces the domain of a variable X to make an arc ⟨X,c⟩ arc consistent, which arcs does it need to reconsider?



AC does not need to reconsider other arcs

- If arc ⟨Y,c⟩ was arc consistent before, it will still be arc consistent.
  "Consistent before" means each element y_i in Y must have an element x_i in X that satisfies the constraint. Those x_i would not be pruned from Dom(X), so arc ⟨Y,c⟩ stays consistent

- If an arc ⟨X,c_i⟩ was arc consistent before, it will still be arc consistent

The domains of Z_i have not been touched

- Nothing changes for arcs of constraints not involving X

Which arcs need to be reconsidered?

- Consider the arcs in turn, making each arc consistent
- Reconsider arcs that could be made inconsistent again by this pruning

- DO trace on ‘simple problem 1’ and on
  ‘scheduling problem 1’, trying to predict
- which arcs are not consistent and

- which arcs need to be reconsidered after each removal in

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

TDA:
ToDoArcs,
blue arcs
in AIspace

Scope of constraint c is

Local

Output

the set of variables involved in that constraint

arc-consistent domains for each variable

1:
2:
3:

D_X is a set of values for each variable X

TDA is a set of arcs

X's domain changed:

for each variable X do

⇒ arcs (Z,c') for variables Z sharing a constraint c' with X could become inconsistent, thus are

$D_X \leftarrow$ dom(X)

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

ND_X: values x for X for

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this

4: while (TDA $\neq$ {}) which there is a value for added to TDA

5: select $\langle X,c\rangle \in$TDA y supporting x 6: TDA $\leftarrow$TDA \ $\{\langle X,c\rangle\}$

7: $ND_X \leftarrow \{x \mid x \in D_X$ and $\exists\, y \in D_Y$ s.t. (x, y) satisfies $c\}$

8: if $(ND_X \neq D_X)$ then

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

If arc was inconsistent

10: $D_X \leftarrow ND_X$

Domain is reduced

11: return $\{D_X \mid X$ is a variable$\}$

procedure (compare with DFS)

- let the max size of a variable domain be d
- let the number of variables be n
- Worst-case time complexity of Backtracking (DFS with pruning?)

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
  - Worst-case time complexity of Backtracking (DFS

Arc Consistency Algorithm: Complexity

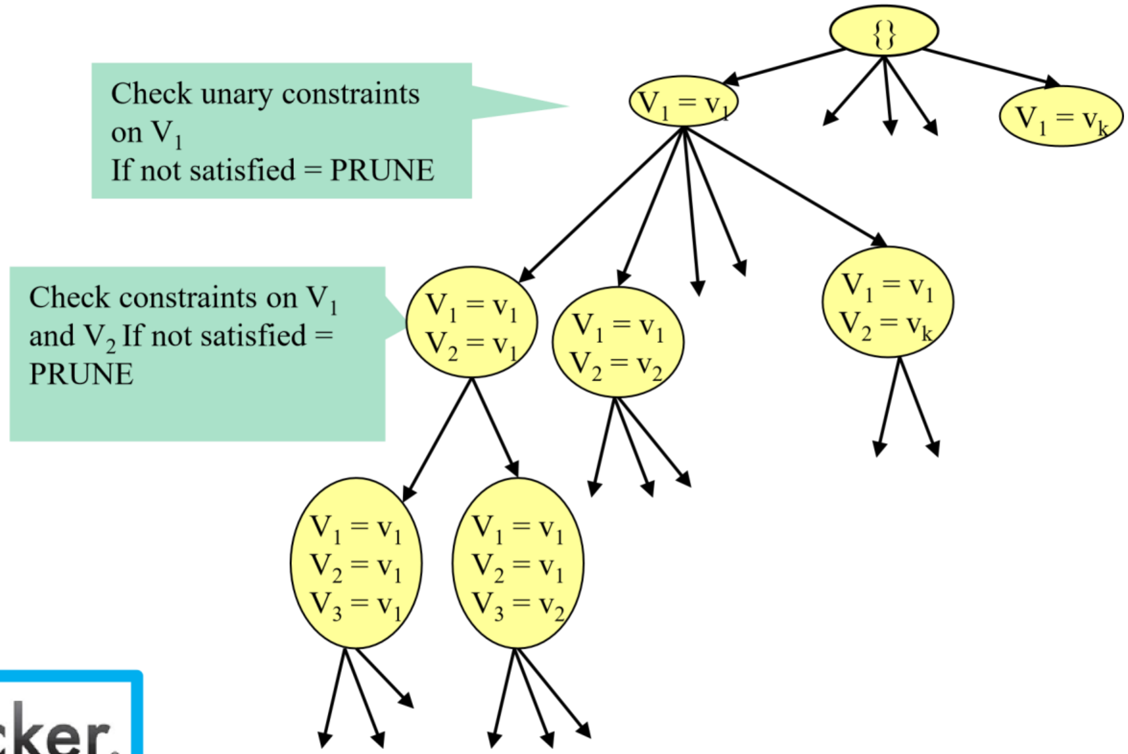
- Let's determine Worst-case complexity of this
  with pruning?)

A. $O(n*d)$

B. $O(d^n)$

C. $O(n^d)$

D. $O(n * d^2)$

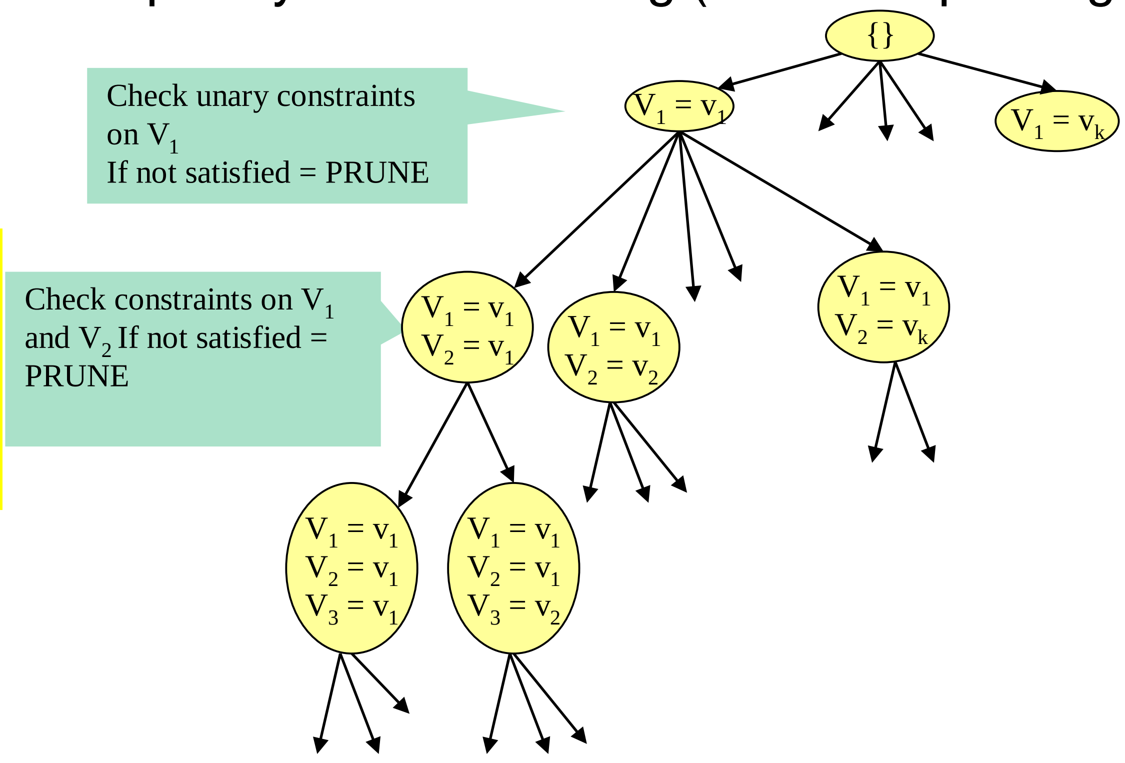


procedure (compare with DFS)

- let the max size of a variable domain be d

Arc Consistency Algorithm: Complexity

- Let’s determine Worst-case complexity of this
  - let the number of variables be n
  - Worst-case time complexity of Backtracking (DFS with pruning?)

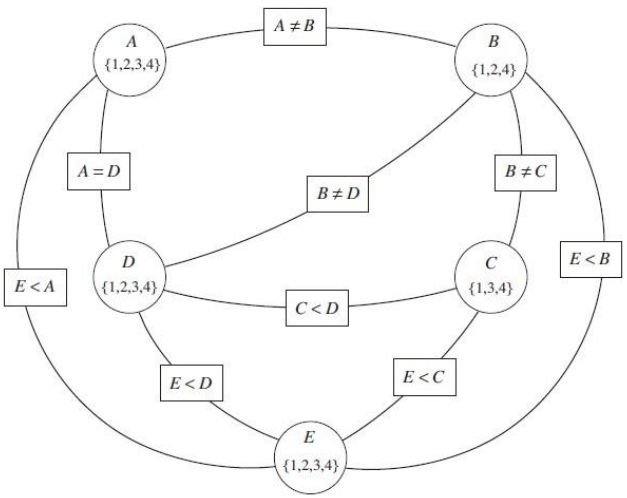


B. $O(d^n)$

It is DFS!

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS……$O(d^n)$……..)
  - 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

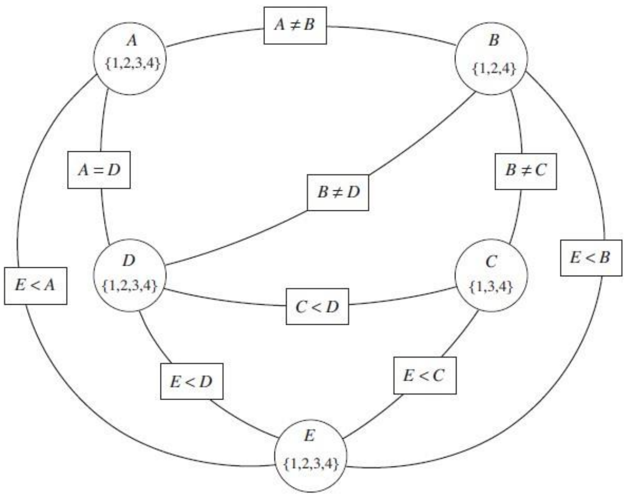
- Let's determine Worst-case complexity of this procedure

(compare with DFS $O(d^n)$)

- 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, at worst, the same arc can be inserted in the ToDoArc list?

Arc Consistency Algorithm: Complexity

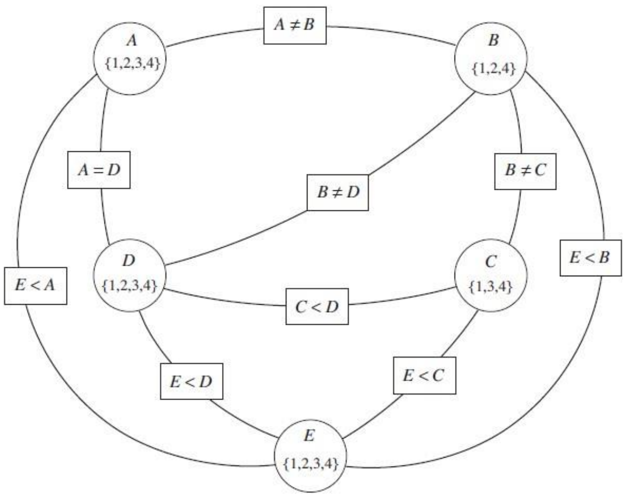


- Let's determine Worst-case complexity of this procedure (compare with DFS $O(d^n)$)
  - let the max size of a variable domain be d

Arc Consistency Algorithm: Complexity

•

- let the number of variables be n
- The max number of binary constraints is ? (n * (n-1)) / 2
- How many times, at worst, the same arc can be inserted in the ToDoArc list? O(d)



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

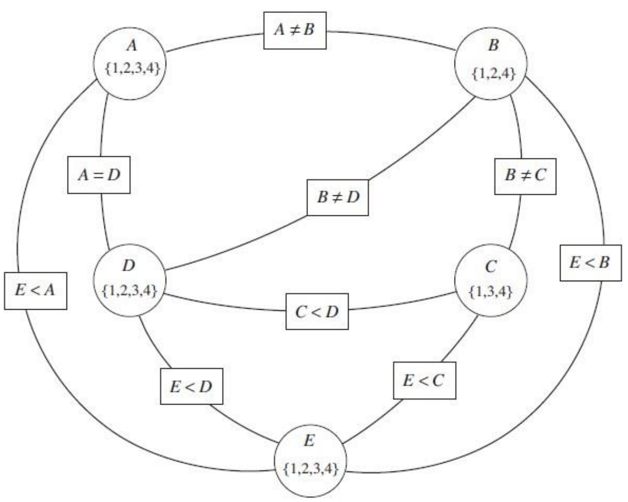
A. $O(n^2)$ B. O(d) C. O(n * d) 2

D. O(d)

Arc Consistency Algorithm: Complexity

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

- 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, at worst, the same arc can be inserted in the ToDoArc list? O(d)
- How many steps are involved in checking the consistency of an arc? $O(d^2)$



- Overall complexity: $O(n^2d^3)$

Arc Consistency Algorithm: Complexity

-
    - Compare to $O(d^N)$ of DFS. Arc consistency is MUCH faster

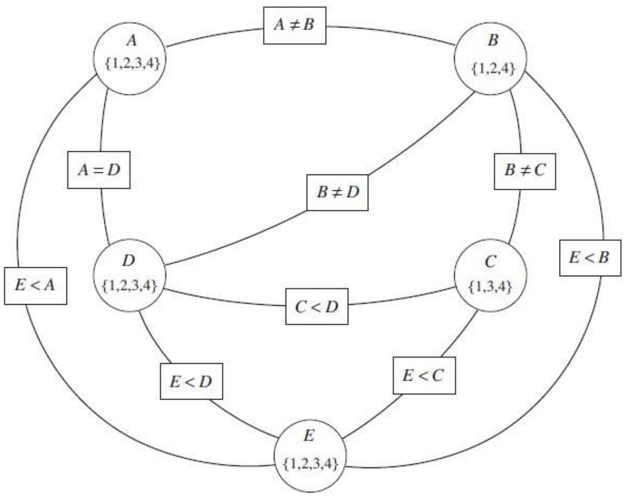
So did we find a polynomial algorithm to solve CPSs?

50

Arc Consistency Algorithm: Complexity

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

  - 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, at worst, the same arc can be inserted in the ToDoArc list? O(d)
  - How many steps are involved in checking the consistency of an arc? $O(d^2)$
  - Overall complexity: $O(n^2d^3)$

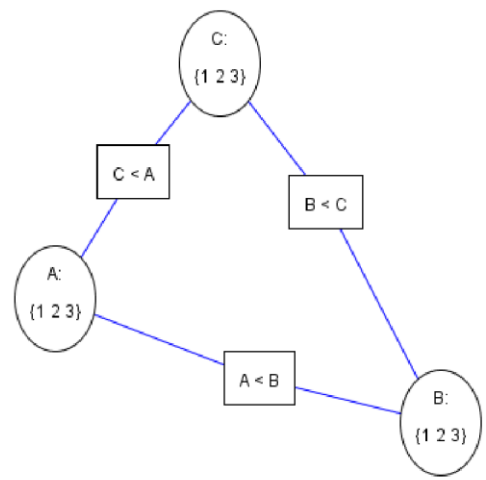


- Compare to $O(d^N)$ of DFS. Arc consistency is MUCH faster So did we find a polynomial algorithm to solve CPSs?

No, AC does not always solve the CPS. It is a way to possibly simplify the original CSP and make it easier to solve

Arc Consistency Algorithm: Interpreting Outcomes

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



- Each domain has a single value,
  - ✓ e.g. built-in AISpace example “Scheduling problem 1” ✓ We have: a (unique) solution.

- At least one domain is empty,
  - ✓ We have: No solution! All values are ruled out for this variable.

- ✓ e.g. try this graph (can easily generate it by modifying Simple Problem 2)

- Some domains have more than one value,

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

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

YES

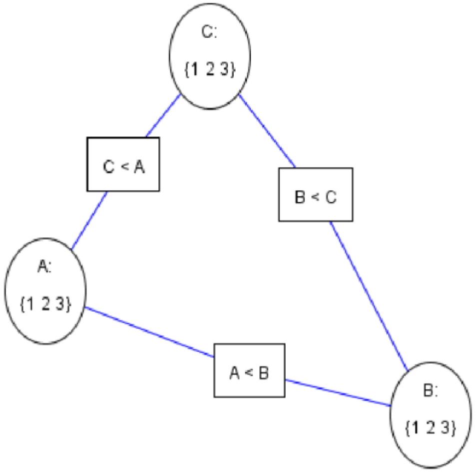
- Example: vars A, B, C with domain {1, 2} and constraints $A \neq B$, $B \neq C$, $A \neq C$
- Or see AIspace CSP applet Simple Problem 2

Arc Consistency Algorithm: Interpreting Outcomes

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

- Each domain has a single value,
  - ✓ e.g. built-in AISpace example “Scheduling problem 1”
  - ✓ We have: a (unique) solution.

- At least one domain is empty,
  - ✓ We have: No solution! All values are ruled out for this variable.
  - ✓ e.g. try this graph (can easily generate it by modifying Simple Problem 2)



- Some domains have more than one value,
  - ✓ There may be: one solution, multiple ones, or none

✓ Need to solve this new CSP (usually simpler) problem:

- same constraints, domains have been reduced

Learning Goals for CSP

- Define possible worlds in term of variables and their domains
- Compute number of possible worlds on real examples
- Specify constraints to represent real world problems differentiating between:
- Unary and k-ary constraints
- List vs. function format
- Verify whether a possible world satisfies a set of constraints (i.e., whether it is a model, a solution)
- Implement the Generate-and-Test Algorithm. Explain its disadvantages.

- Solve a CSP by search (specify neighbors, states, start state, goal state). Compare strategies for CSP search. Implement pruning for DFS search in a CSP.
- Define/read/write/trace/debug the arc consistency algorithm. Compute its complexity and assess its possible outcomes