Constraint Satisfaction Problems (CSPs)

Computer Science cpsc322, Lecture 4

(Textbook Chpt 4.0 – 4.6)

May, 17, 2012

Announcements

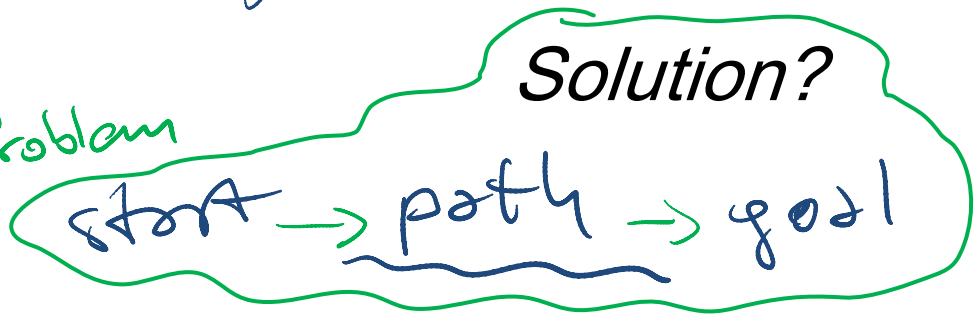
- **Search wrap-up**
  - Go back to learning goals (end of slides)
  - Make sure you understands the inked slides
  - More details or different examples on textbook
  - Work on the practice exercises
  - Look at assignment 1 solutions
  - If still confused, come to office hours

- **MIDTERM: Mon May 28th – 3PM (room TBA)**

Standard Search

To learn about **search** we have used it as the *reasoning strategy* for a **simple goal-driven planning agent**.....

vacuum-cleaner
8-puzzle
Delivery-robot
Marina-Problem

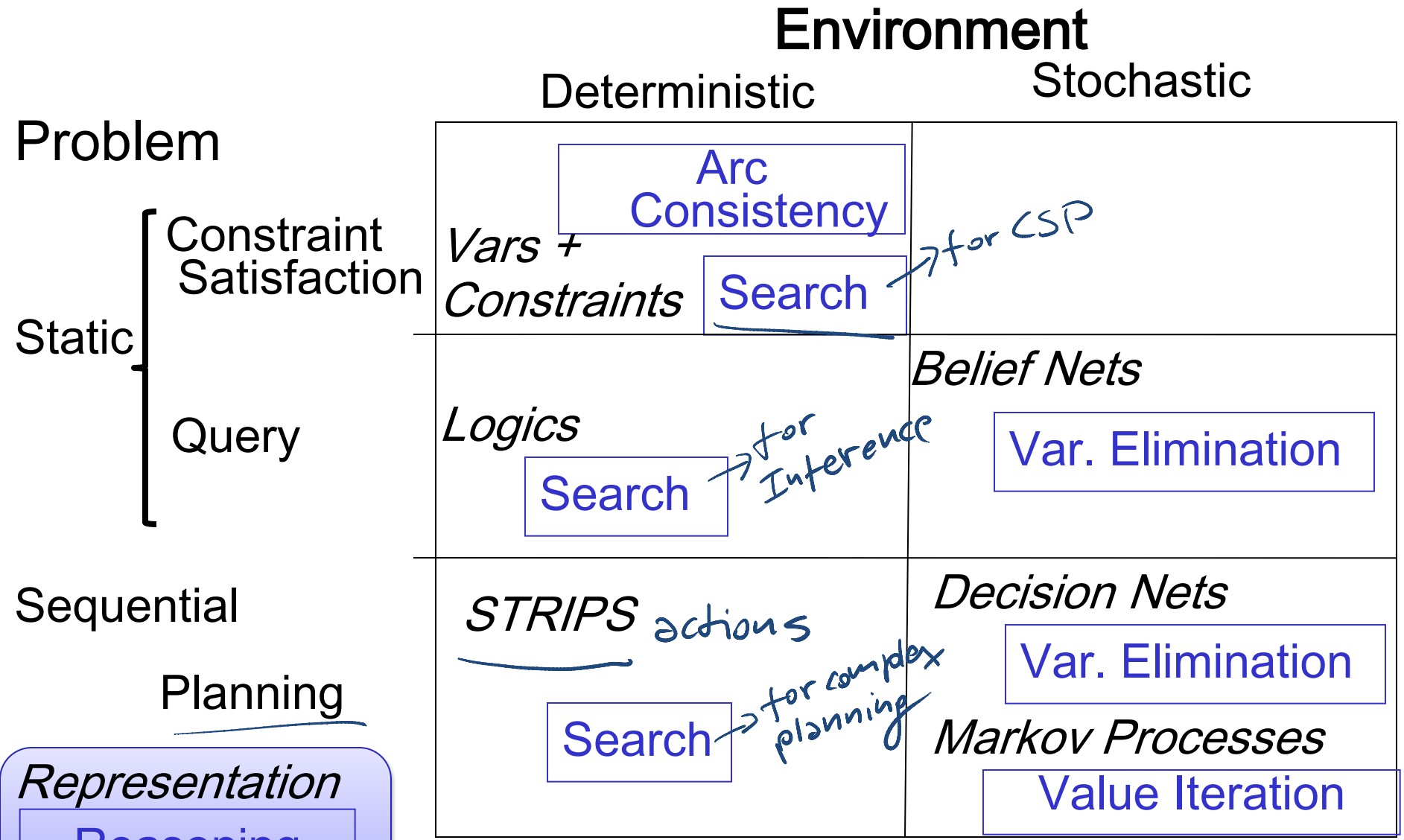


Standard search problem: An agent can solve a problem by searching in a space of states

- state is a "black box" – any arbitrary data structure that supports **three problem-specific routines**

① successors / neighbors(n) ② heuristic(n) ③ goal(n)

Modules we'll cover in this course: R&Rsys



Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

- State
- Successor function
- Goal test
- Solution

this lecture

Planning :

- State
- Successor function
- Goal test
- Solution

Inference

- State
- Successor function
- Goal test
- Solution

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting

Variables/Features, domains and Possible Worlds

- Variables / features
  - we denote variables using capital letters
  - each variable V has a domain *dom*(V) of possible values

A, B

dom(B) = dom(A) = {0, 1}

- Variables can be of several main kinds:
  - Boolean: |*dom*(V)| = 2
  - Finite: the domain contains a finite number of values
  - ~~Infinite but Discrete: the domain is countably infinite~~
  - ~~Continuous: e.g., real numbers between 0 and 1~~

propositions

not in this course

- Possible world: a complete assignment of values to a set of variables

e.g. {A = 1, B = 0}

Example (lecture 2)

Mars Explorer Example

Weather	{S, C}
Temperature	[-40, 40]
Longitude	[0, 359]
Latitude	[0, 179]

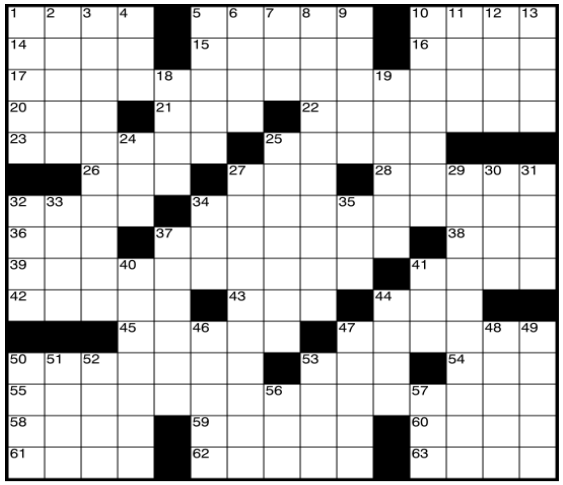
One possible world (state) {S, -30, 320, 91}

Number of possible (mutually exclusive) worlds (states)

2 x 81 x 360 x 180

How many possible worlds?

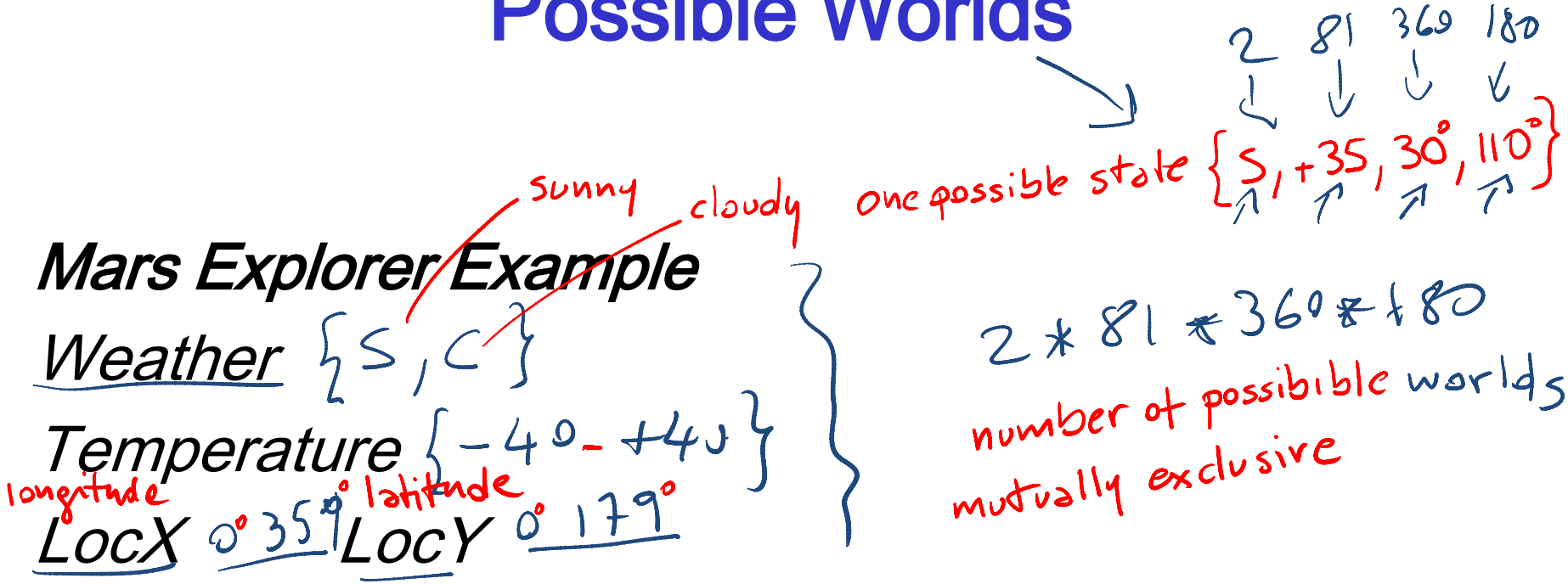
- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - possible worlds: all ways of assigning letters to cells



- Number of empty cells?
- Number of letters in the alphabet?
- How many possible worlds? (assume any combination is ok)

193*26 193^{26} 26^{193}

Possible Worlds

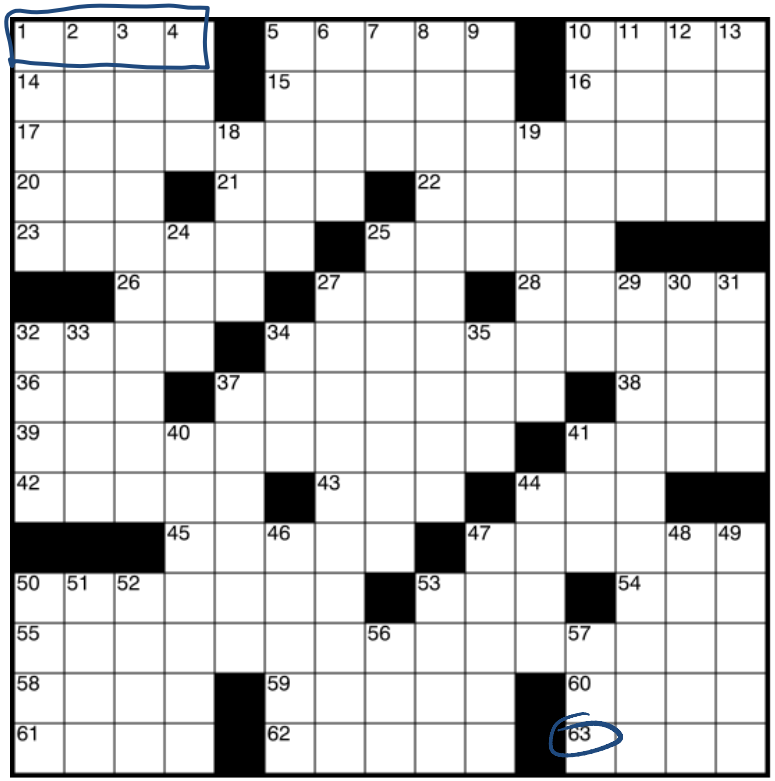


Product of cardinality of each domain

… always exponential in the number of variables

Examples

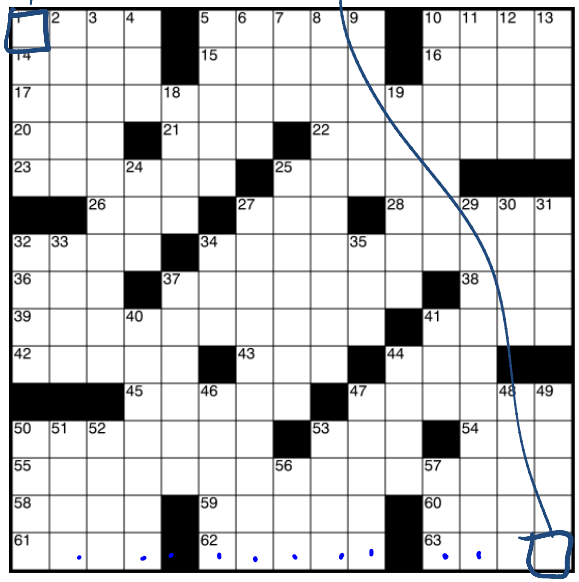
- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are valid English words of required length
  - possible worlds: all ways of assigning words



- *Number of English words?*
- *Number of words of length k?*
- *So, how many possible worlds?*

More Examples

- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - possible worlds: all ways of assigning letters to cells
    - *So, how many possible worlds?*

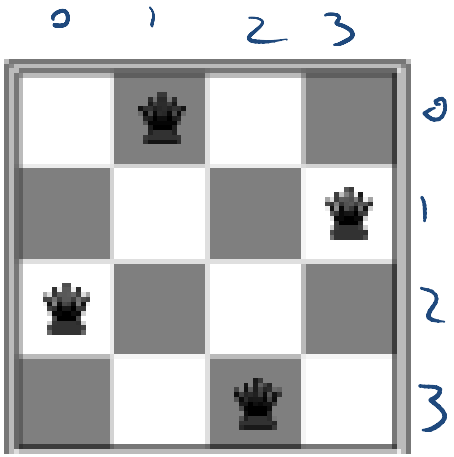


- Sudoku:
  - variables are empty cells
  - domains are numbers between 1 and 9
  - possible worlds: all ways of assigning numbers to cells
- *So, how many possible worlds?*

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

More examples

- n-Queens problem
  - variable: location of a queen on a chess board
    - there are *n* of them in total, hence the name
  - domains: grid coordinates n^2
  - possible worlds: locations of all queens



$\frac{16!}{12!\,4!}$

$(n^2)^n$

no overlaps

$\binom{n^2}{n} = \frac{n^2!}{(n^2-n)!\,n!}$

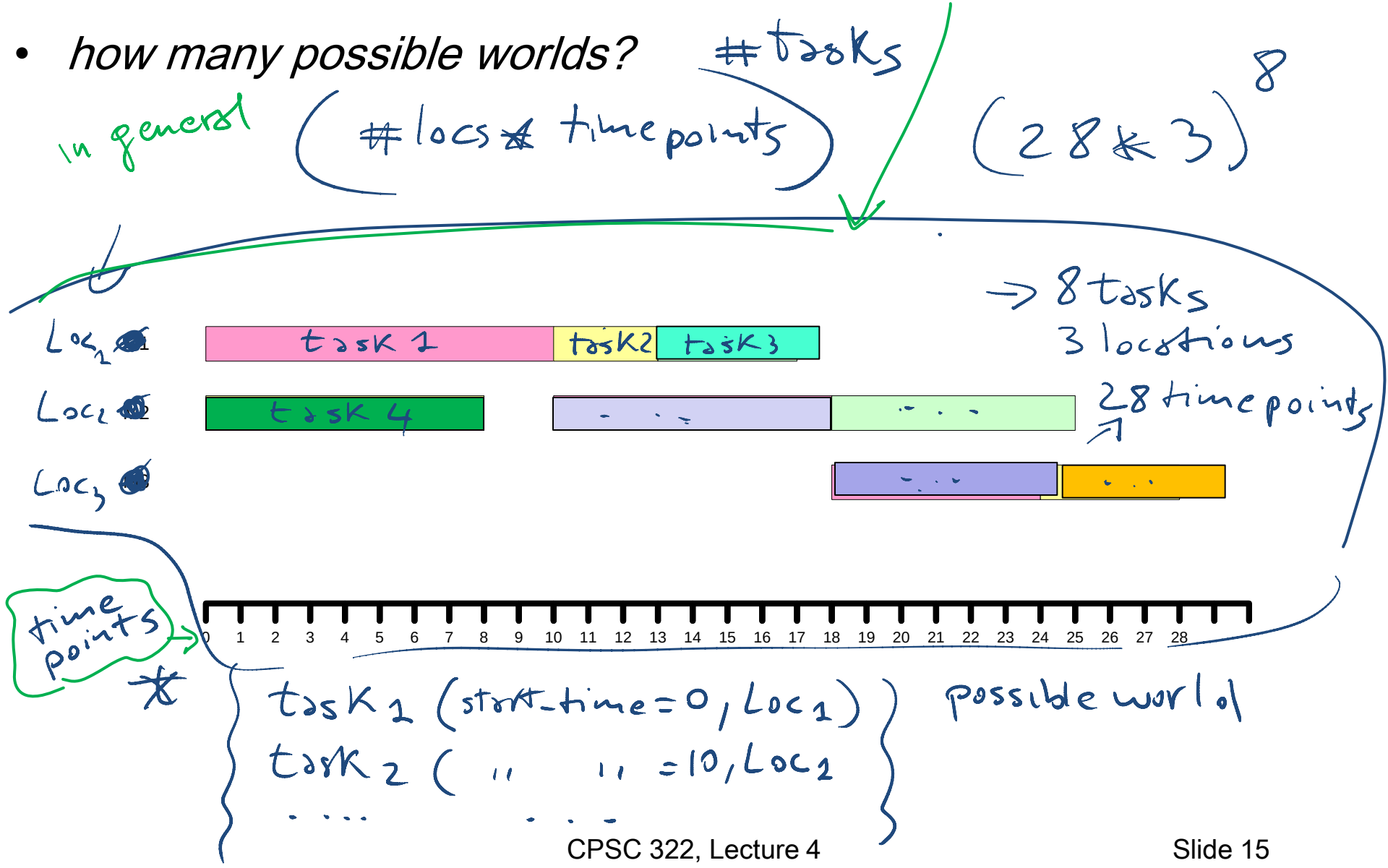
possible ways to choose n location out of n^2.

More examples

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - possible worlds: time/location assignments for each task

Scheduling possible world

- *how many possible worlds?*



More examples….

- Map Coloring Problem
  - variable: regions on the map
  - domains: possible colors
  - possible worlds: color assignments for each region

- *how many possible worlds?*

(#colors)^#regions

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting

Constraints

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 the domains of k different variables
  - it turns out that k-ary constraints can always be represented as binary constraints, so we'll *mainly* only talk about this case

- **Constraints can be specified by**
  - giving a function that returns true when given values for each variable which satisfy the constraint
  - giving a list of valid domain values for each variable participating in the constraint

Example: Map-Coloring

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

Domains D_i = {red,green,blue}

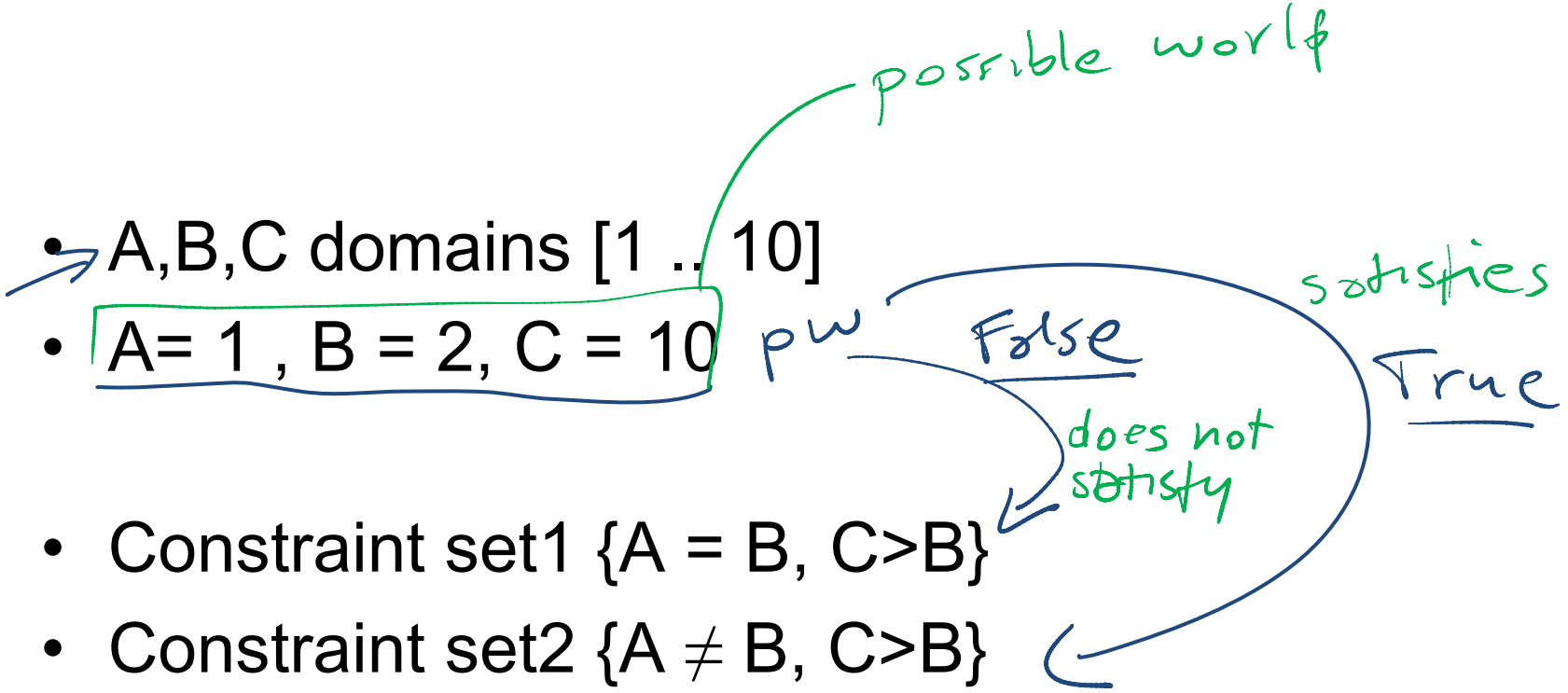
Constraints: adjacent regions must have different colors

e.g., WA ≠ NT, SA≠NT, SA≠WA . . . - - -

or, NT(WA,NT) in {(red,green),(red,blue),(green,red),
(green,blue),(blue,red),(blue,green)}

Constraints (cont.)

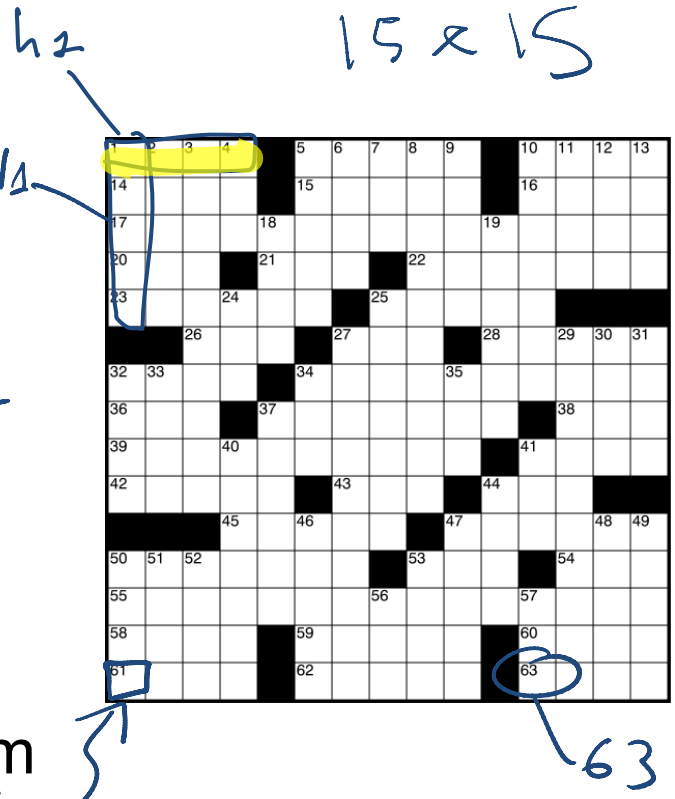
- A **possible world** satisfies **a set of constraints** if the set of variables involved in each constraint take values that are consistent with that constraint



Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are valid English words
  - *constraints:* words have the same letters at points where they intersect

$h_1[0] = V_1[0]$, ... ≈ 225 constraints



- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - *constraints:* sequences of letters form valid English words

A

63 constraints

concatenate(A[0,0] ... A[0,3]) ∈ English word of length 4

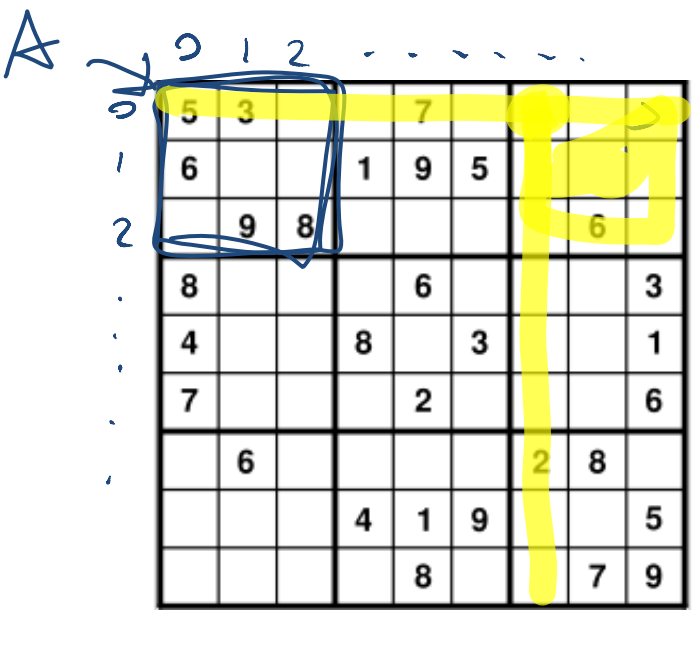
Examples

actually 20 because some overlap

- Sudoku:
  - variables are cells
  - domains are numbers between 1 and 9
  - *constraints:* rows, columns, boxes contain all different numbers

# of constraint =

$\sim$ #empty-cells $\times$ 24 $-\{8+8+8\}$



for all i, $i \neq 2$

$A[2,0] \neq A[i,0]$

which means

$A[2,0] \neq A[0,0]$

$A[2,0] \neq A[1,0]$

solution

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

More examples

- n-Queens problem
  - variable: location of a queen on a chess board
    - there are *n* of them in total, hence the name
  - domains: grid coordinates
  - *constraints*: no queen can attack another

eg two queens cannot be on the same column / row

$Q_1 = \{x_1, y_1\}$

$Q_2 = \{x_2, y_2\}$

$x_1 = x_2$ and $y_1 = y_2$

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - *constraints*: e.g. $task_1(loc_1, start\text{-}t_1)$ if $start\text{-}t_1 = start\text{-}t_2$ $task_2(loc_2, start\text{-}t_2)$ then $loc_1 \neq loc_2$
    - ✓ tasks can't be scheduled in the same location at the same time;
    - ✓ certain tasks can be scheduled only in certain locations;
    - ✓ some tasks must come earlier than others; etc.

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting

Constraint Satisfaction Problems: definitions

Definition (Constraint Satisfaction Problem)

A constraint satisfaction problem consists of

- a set of variables
- a domain for each variable
- a set of constraints

Definition (model / solution)

A model of a CSP is an assignment of values to variables that satisfies all of the constraints.

possible worlds

Constraint Satisfaction Problems: definitions

Definition (Constraint Satisfaction Problem)

A constraint satisfaction problem consists of

- a set of variables
- a domain for each variable
- a set of constraints

possible worlds
3 * 5 * 5

A, B, C

dom C = {1, 2, 3}

dom A = {1, 2, 3, 4, 5} = dom(B)

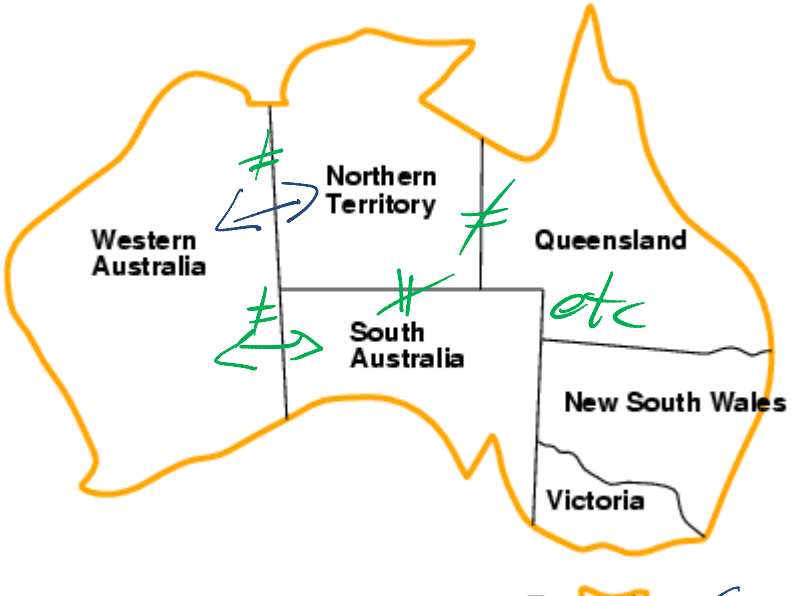
B = 5 C > B A = B no sol
" C < B "

Definition (model / solution)

A model of a CSP is an assignment of values to variables that satisfies all of the constraints.

B = 5 A = 5 C = 1
" " " C = 2
" " " C = 3

Example: Map-Coloring



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

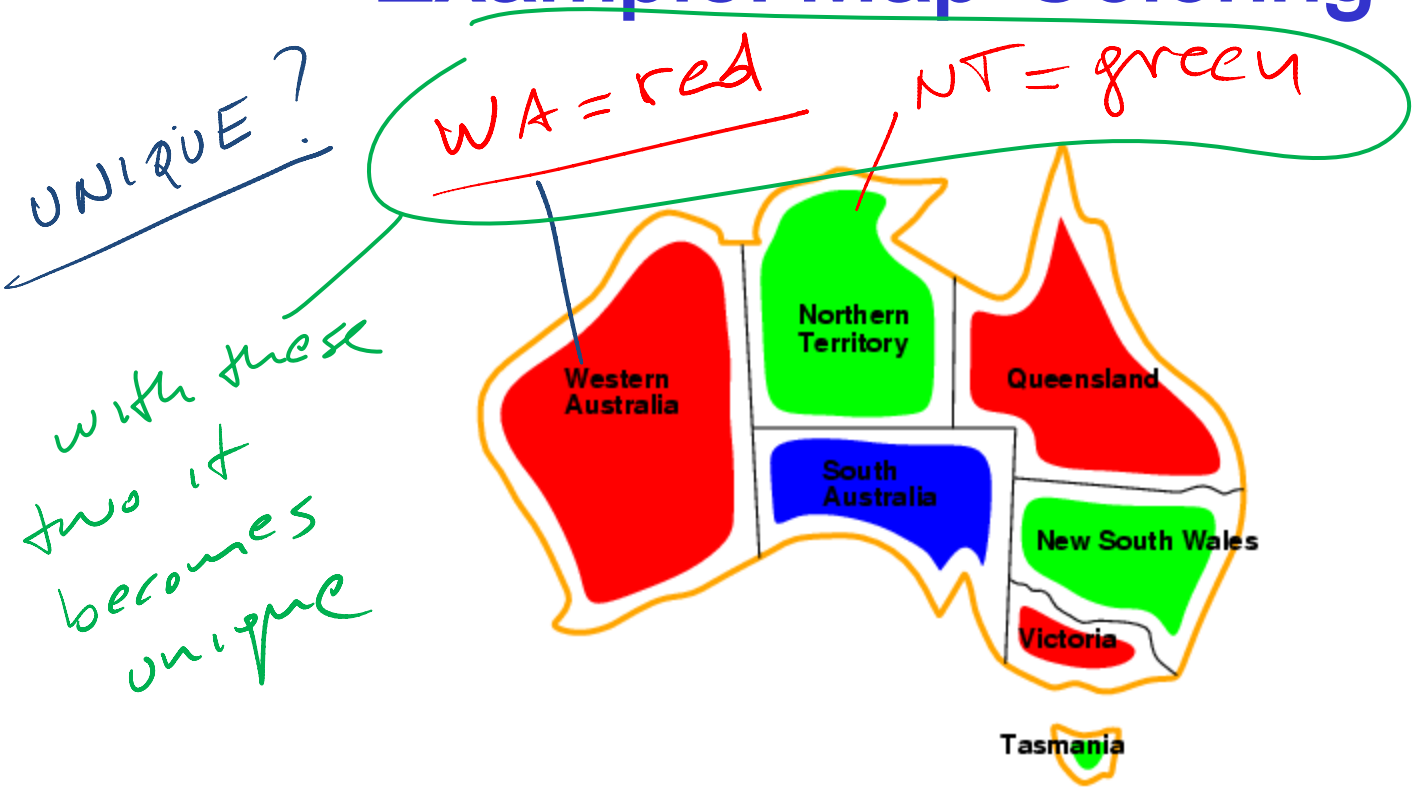
Domains D_i = {red,green,blue}

Constraints: adjacent regions must have different colors

e.g., WA ≠ NT, or . . . .

(WA,NT) in {(red,green),(red,blue),(green,red),
(green,blue),(blue,red),(blue,green)}

Example: Map-Coloring



Models / Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red,SA = blue, T = green

Constraint Satisfaction Problem: Variants

We may want to solve the following problems using a CSP

A. determine whether or not a model exists

B. find a model

C. find all of the models

D. count the number of the models

E. find the best model given some model quality

   - this is now an optimization problem

F. determine whether some properties of the variables hold in all models

Learning Goals so far….

- 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)

User Study Participants Needed

What:

- Use a prototype tool for updating a database
- 60-90 minutes duration

Why:

- Help us evaluate our research
- Get a $10 Starbucks gift card

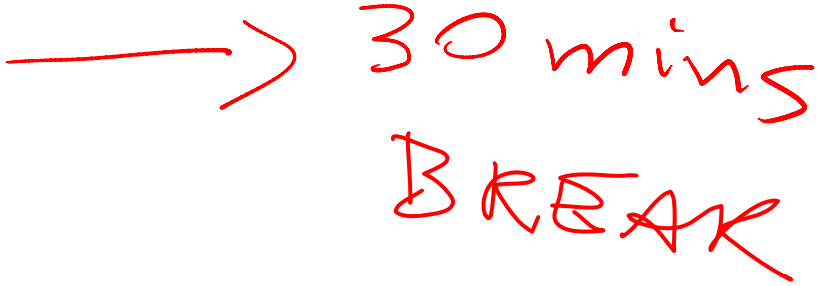
Who: Anyone who has taken CPSC 304 or understands basic database concepts

Where: ICCS Building

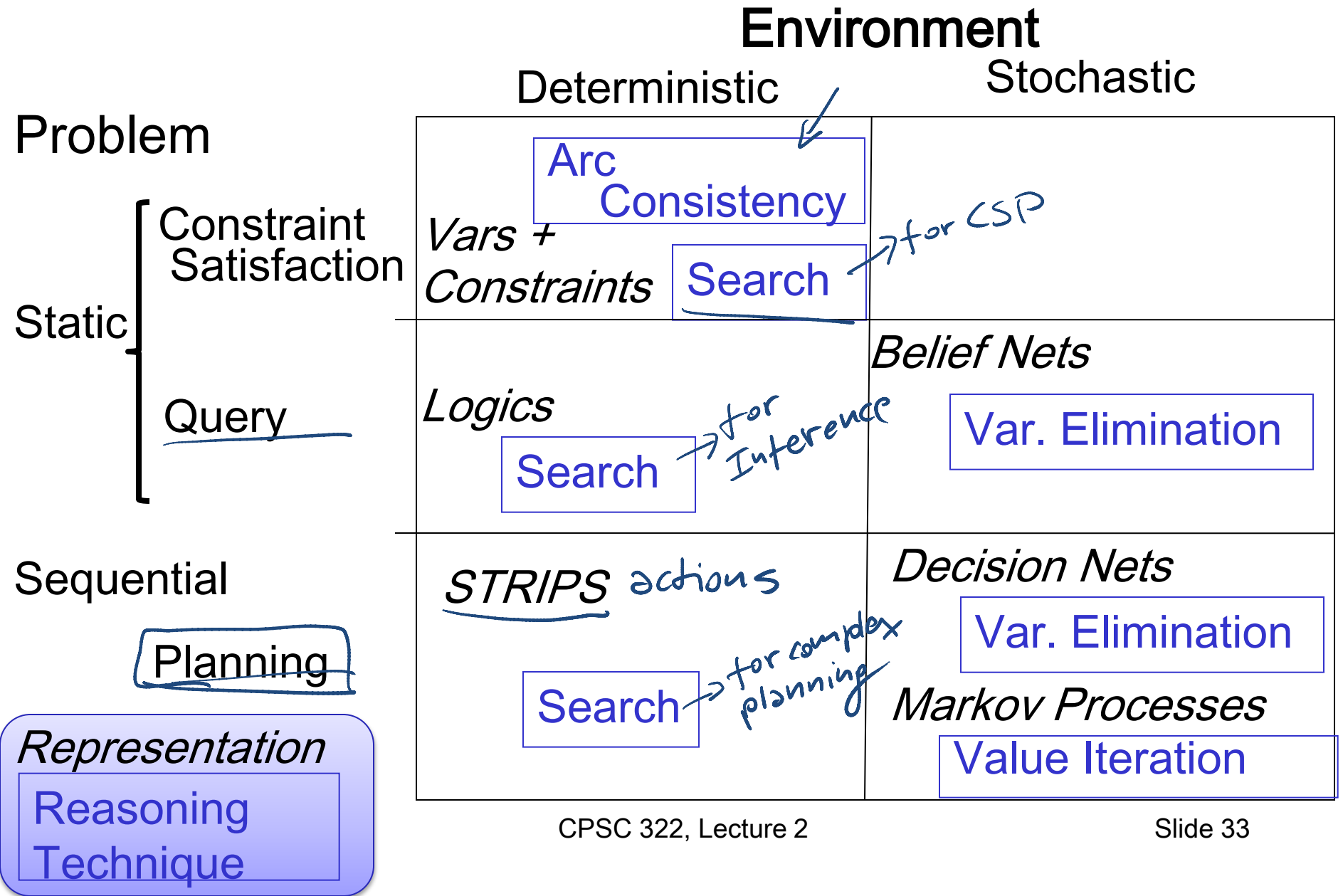
Contact *Michael Lawrence*, mklawren@cs.ubc.ca

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting



Modules we'll cover in this course: R&Rsys



Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

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

Planning :

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

Answering Queries

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

Generate-and-Test Algorithm

- **Algorithm:**
  - Generate possible worlds one at a time
  - Test them to see if they violate any constraints

→ domA = {1, 2, 3, 4, 5}
→ domB = {1, 2, 3, 4, 5}
→ domC = {1, 2, 3}

```
For a in domA
   For b in domB
      For c in domC
      if (a b c) satisfies all constraints
      return (a b c)
return fail
```

- This procedure is able to solve any CSP
- However, the running time is proportional to the number of possible worlds
  - always exponential in the number of variables
  - far too long for many CSPs ☹

Generate and Test (GT) Algorithms

- Systematically check all possible worlds
  - Possible worlds: cross product of domains
    $dom(V_1) \times dom(V_2) \times \cdots \times dom(V_n)$
- Generate and Test:
  - Generate possible worlds one at a time
  - Test constraints for each one.

Example: 3 variables A,B,C

```
For a in dom(A)
    For b in dom(B)
        For c in dom(C)
            if {A=a, B=b, C=c} satisfies all constraints
                return {A=a, B=b, C=c}
fail
```

Generate and Test (GT) Algorithms

- If there are *k* variables, each with domain size *d*, and there are *c* constraints, the complexity of Generate & Test is

$O(ckd)$ $O(ck^d)$ $O(cd^k)$ $O(d^{ck})$

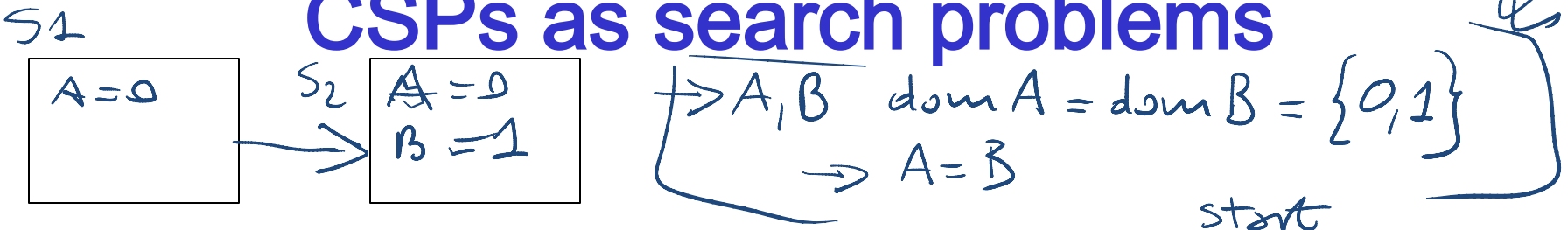
Generate and Test (GT) Algorithms

- If there are *k* variables, each with domain size *d*, and there are *c* constraints, the complexity of Generate & Test is

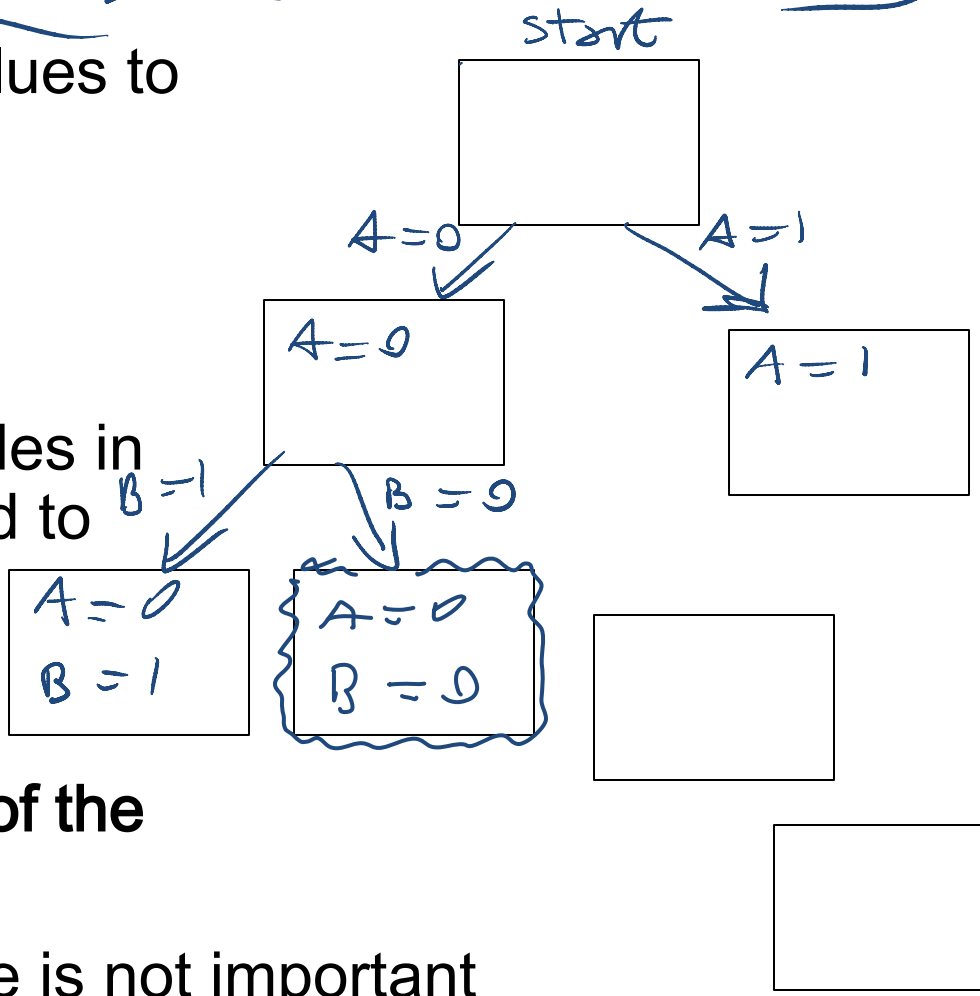
$O(cd^k)$

  - There are d^k possible worlds
  - For each one need to check c constraints

CSPs as search problems



- **states:** assignments of values to a subset of the variables
- **start state:** the empty assignment (no variables assigned values)
- **neighbours** of a state: nodes in which values are assigned to one additional variable
- **goal state:** a state which **assigns a value to each variable**, and **satisfies all of the constraints**



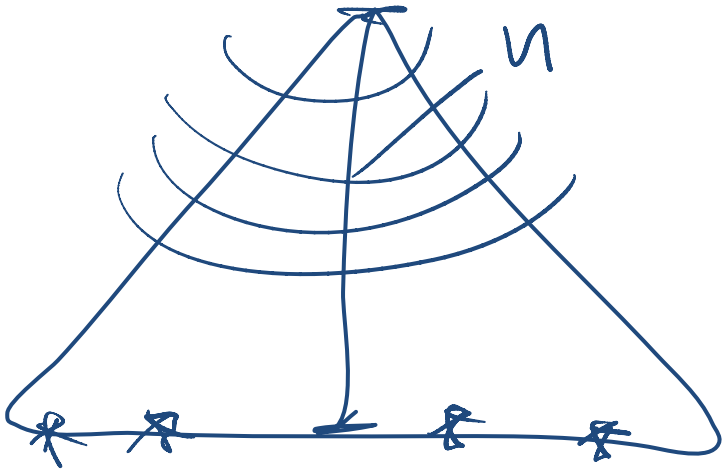
Note: the path to a goal node is not important

CSPs as Search Problems

What search strategy will work well for a CSP?

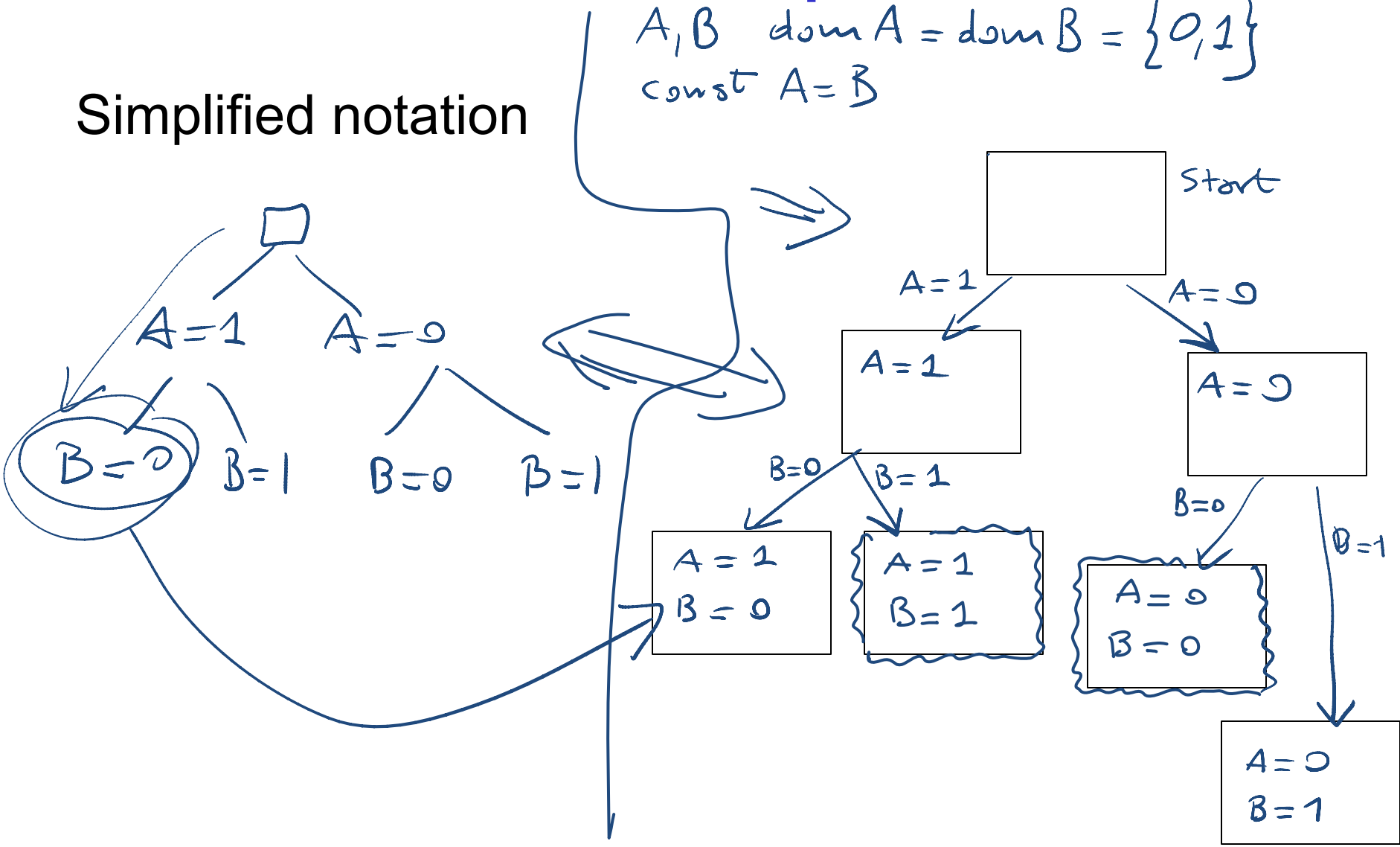
- If there are n variables every solution is at depth n
- Is there a role for a heuristic function?

- the tree is always finite and has no cycles, so which one is better ~~BFS~~ or ~~IDS~~ or DFS?



CSPs as search problems

Simplified notation

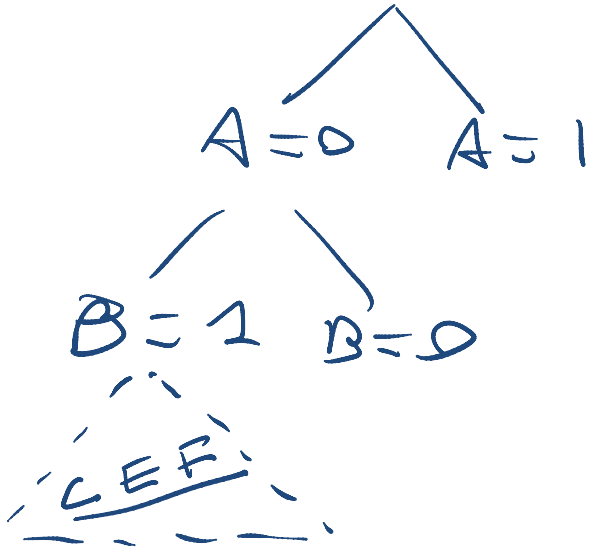


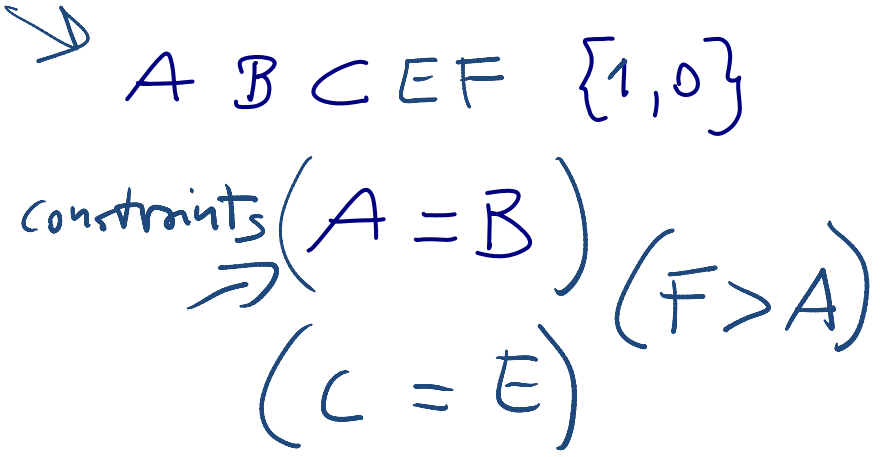
CSPs as Search Problems

How can we avoid exploring some sub-trees i.e.,

prune the DFS Search tree?

- once we consider a path whose end node violates one or more constraints, we know that a solution cannot exist below that point
- thus we should **remove that path** rather than continuing to search

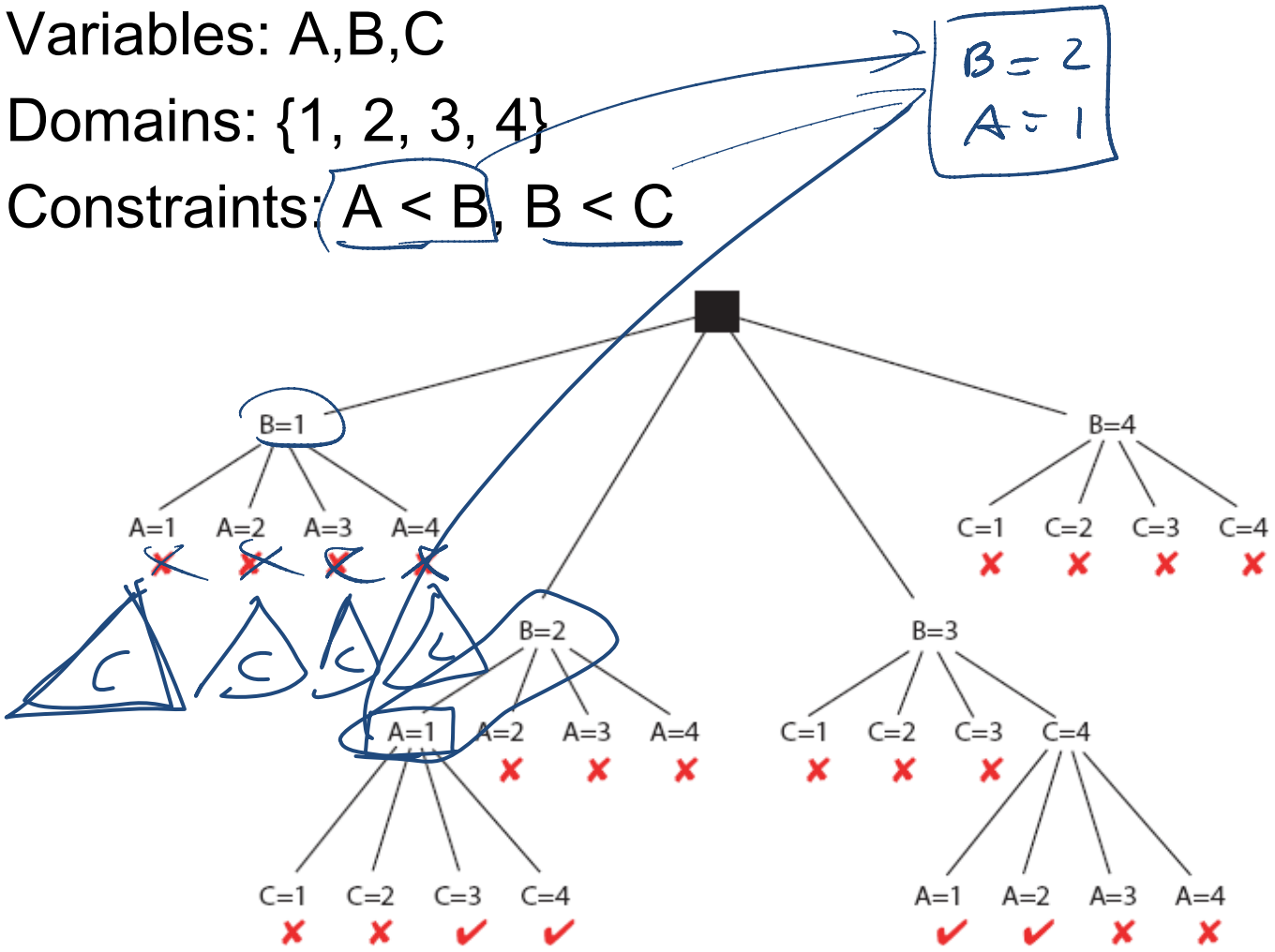




Solving CSPs by DFS: Example

Problem:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C



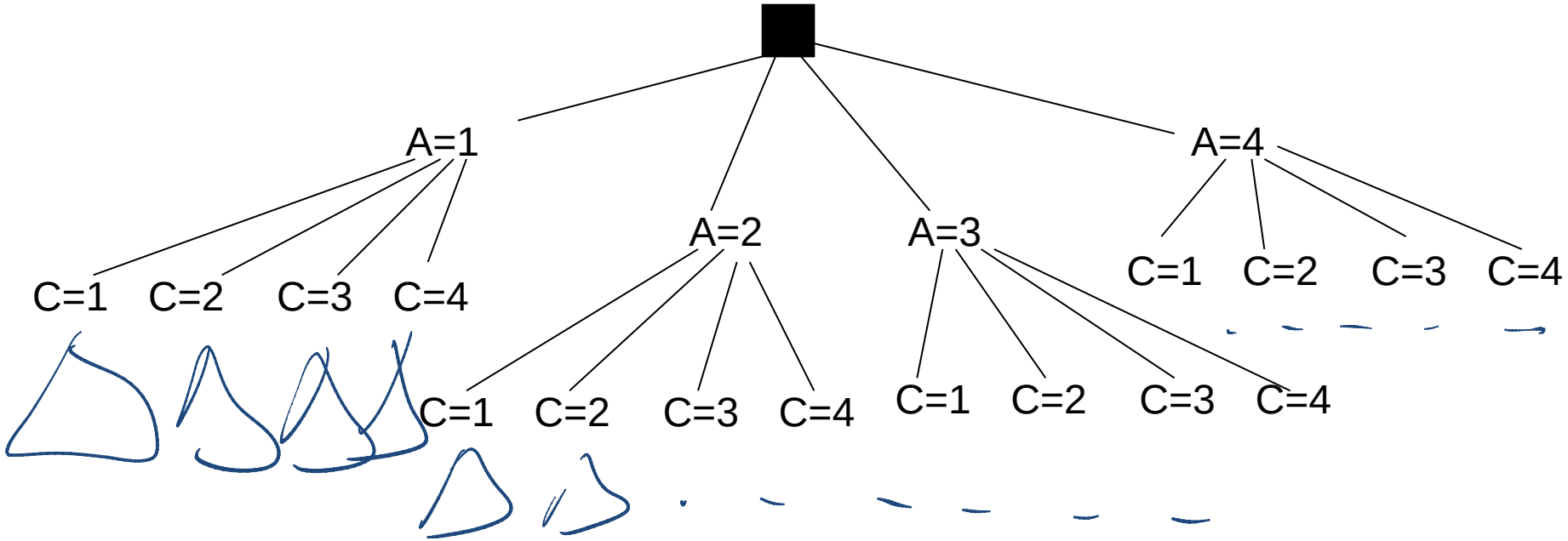
Solving CSPs by DFS: Example Efficiency

Problem:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C

Note: the algorithm's efficiency depends on the order in which variables are expanded

Degree "Heuristics"



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

Inference

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

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting

Can we do better than Search?

Key ideas:

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

Simple when using constraints involving single variables (technically enforcing **domain consistency**)

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

- Example: $\mathbf{D}_B = \{1, 2, 3, 4\}$ is not domain consistent if we have the constraint $B \neq 3$.

How do we deal with constraints involving multiple variables?

Definition (**constraint network**)

A constraint network is defined by a graph, with

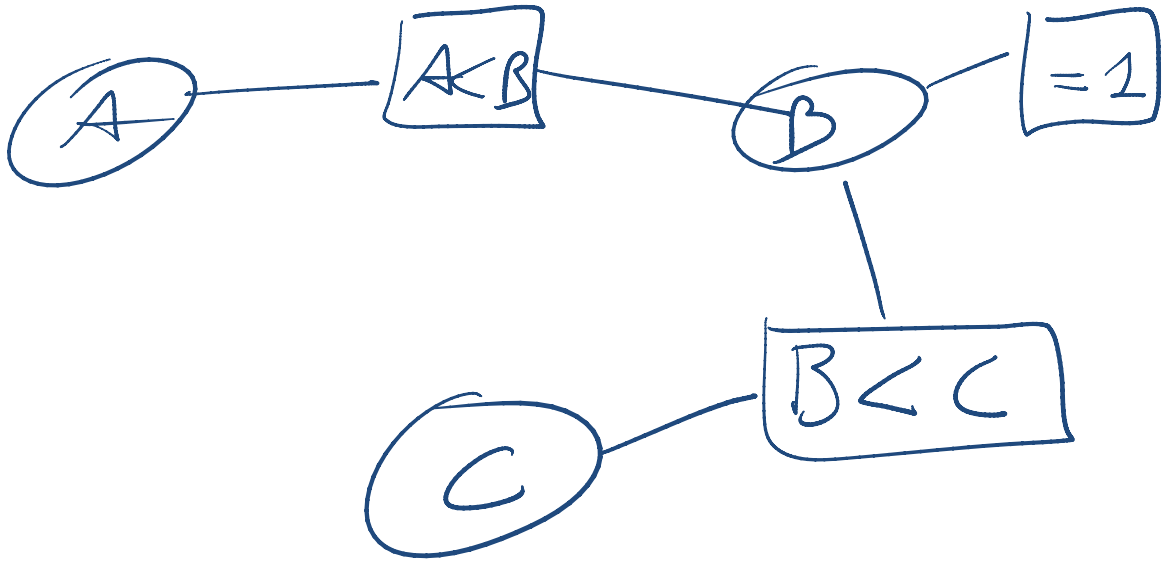
- one **node** for every **variable**
- one **node** for every **constraint**

and undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

A B {0,1}

A = B

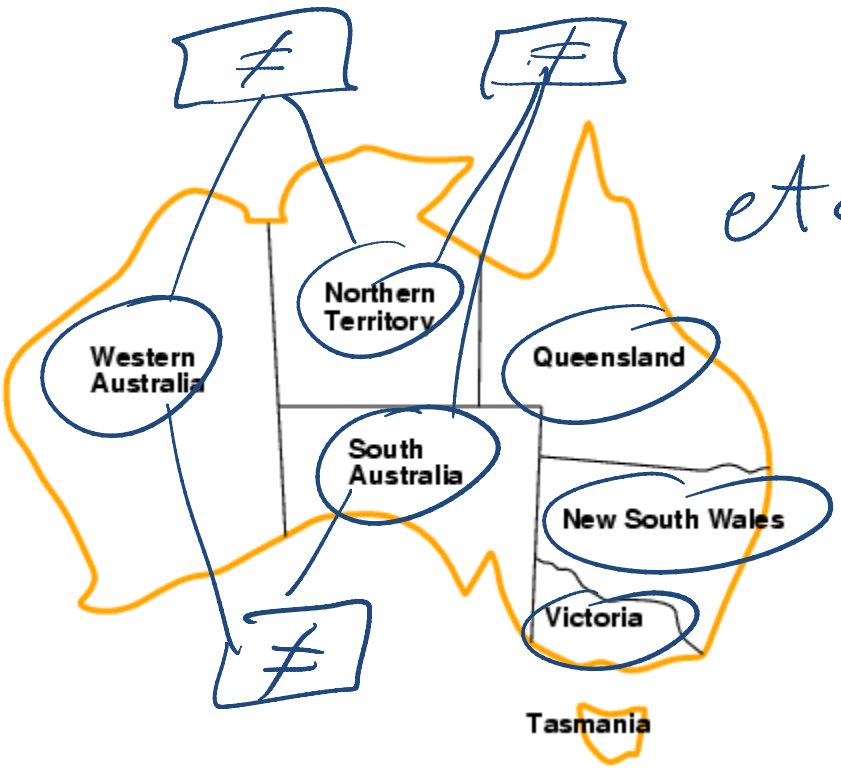
Example Constraint Network



Recall Example:

- Variables: A,B,C
- Domains: {1, 2, 3, 4}
- Constraints: A < B, B < C, B = 1

Example: Constraint Network for Map-Coloring



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

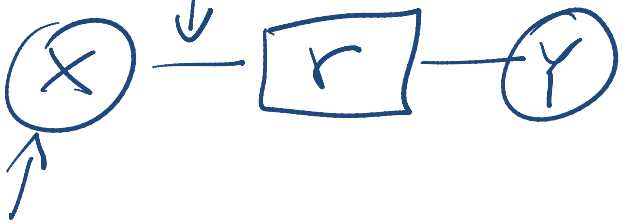
Domains D_i = {red,green,blue}

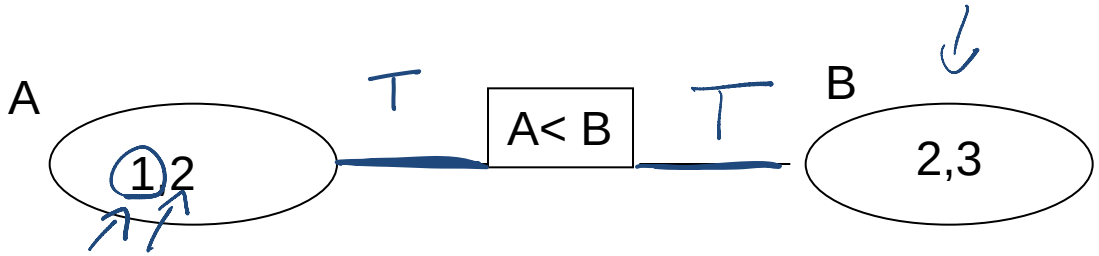
Constraints: adjacent regions must have different colors

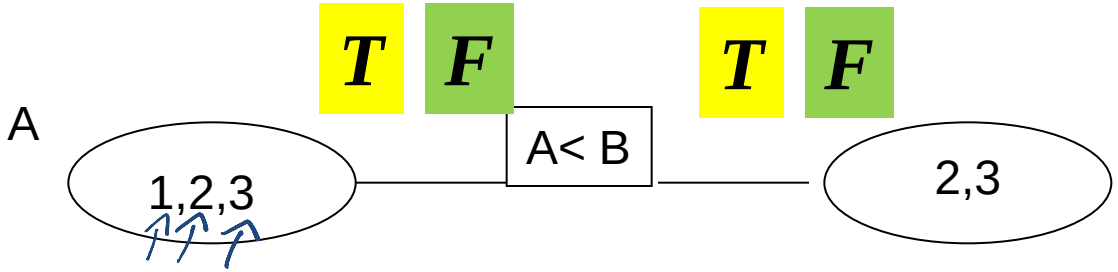
Arc Consistency

Definition (arc consistency)

An arc $\langle X, r(X,Y) \rangle$ 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.

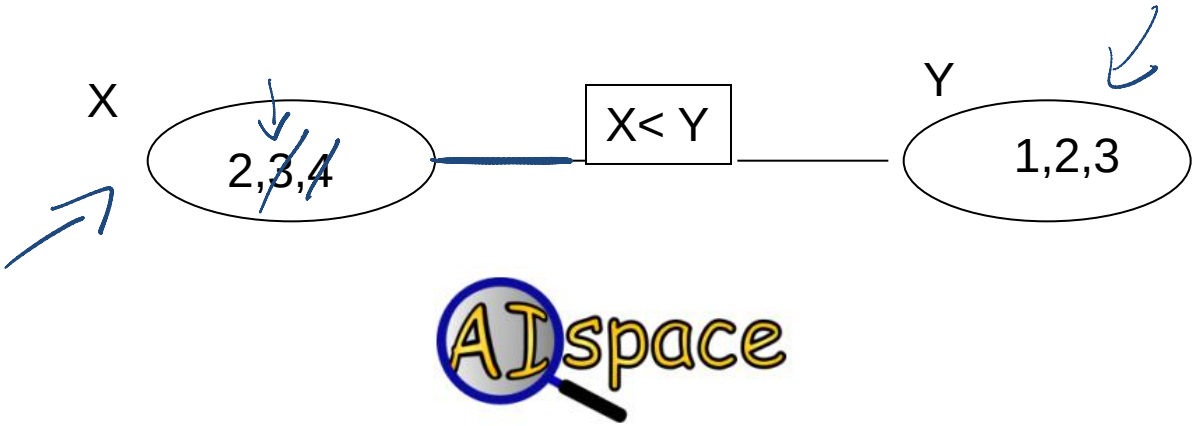






How can we enforce Arc Consistency?

- If an arc $\langle X, r(X,Y) \rangle$ is not arc consistent, all values x in $dom(X)$ for which there is no corresponding value in $dom(Y)$ may be deleted from $dom(X)$ to make the arc $\langle X, r(X,Y) \rangle$ consistent.
  - This removal can never rule out any models/solutions



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

Learning Goals for today's class

You can:

- 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.
- Build a constraint network for a set of constraints.
- Verify whether a network is arc consistent.
- Make an arc arc-consistent.

Lecture Overview

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - Domain Splitting

Arc Consistency Algorithm: high level strategy

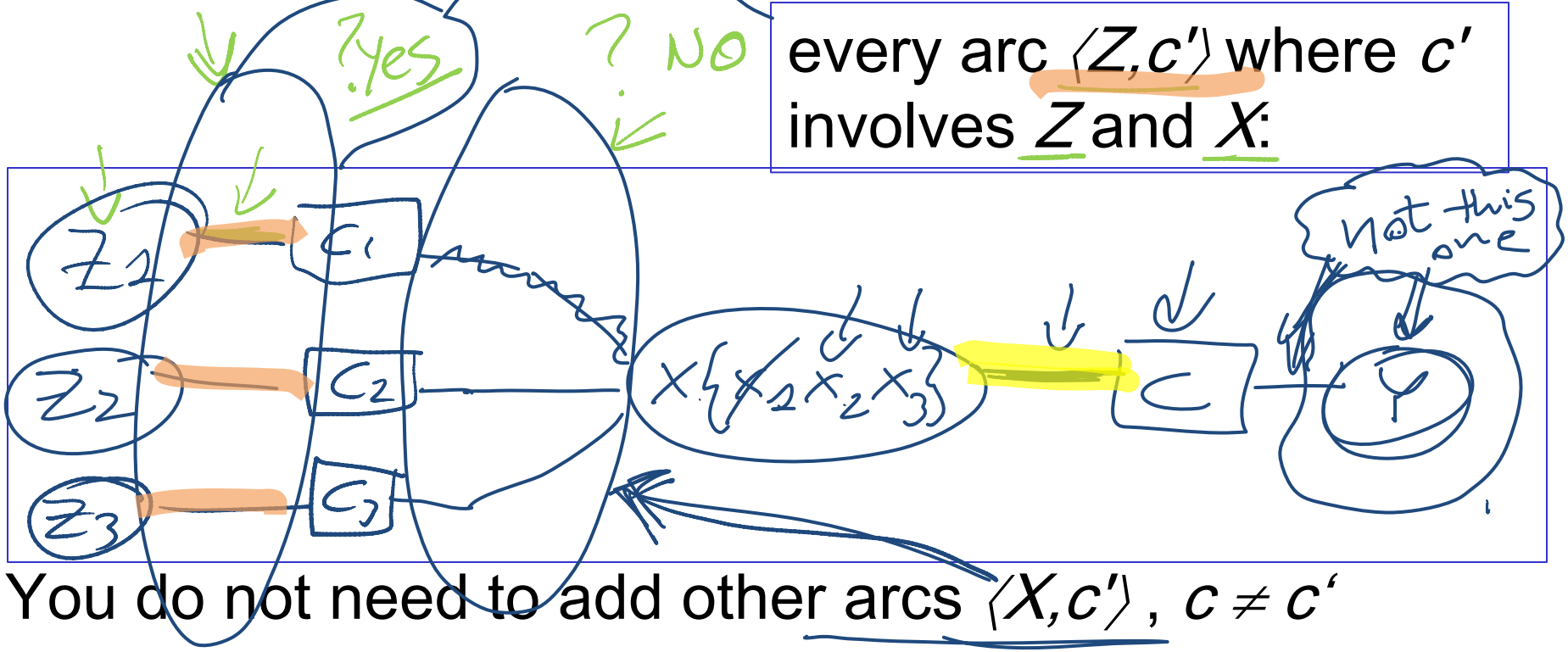
- Consider the arcs in turn, making each arc consistent.
- BUT, arcs may need to be revisited whenever….

- 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

1: **for each** variable X **do**
2: $\mathbf{D}_X \leftarrow$dom(X)
3: TDA $\leftarrow\{\langle X,c\rangle \mid X \in V, c \in C$ and $X \in$ scope(c)$\}$
4: **while** (TDA $\neq \{\}$)
5: **select** $\langle X,c\rangle \in$TDA
6: TDA $\leftarrow$TDA $\setminus \{\langle X,c\rangle\}$
7: $ND_X \leftarrow\{x \mid x \in \mathbf{D}_X$ and $\exists\ y \in \mathbf{D}_Y$ s.t. (x, y) satisfies c$\}$
8: **if** $(ND_X \neq \mathbf{D}_X)$ **then**
9: TDA $\leftarrow$TDA $\cup \{ \langle Z,c'\rangle \mid X \in$ scope(c'), c' $\neq$ c, Z $\in$ scope(c') $\setminus \{X\} \}$
10: $\mathbf{D}_X \leftarrow ND_X$
11: **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

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

X's domain changed: $\Rightarrow$ 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^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

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

$O(d^3 n^2)$ OVERALL COMPLEXITY

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

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

- **Constraint Satisfaction Problems**
  - Variables , Constraints, CSPs
- CSPs Methods
  - Generate-and-Test
  - Search
  - Arc Consistency and AC Algorithm
    - ✓Abstract strategy, Details
    - ✓Complexity, Interpreting the output
  - 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 disjoint cases

- Set of all solution equals to….

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

But what is the advantage?

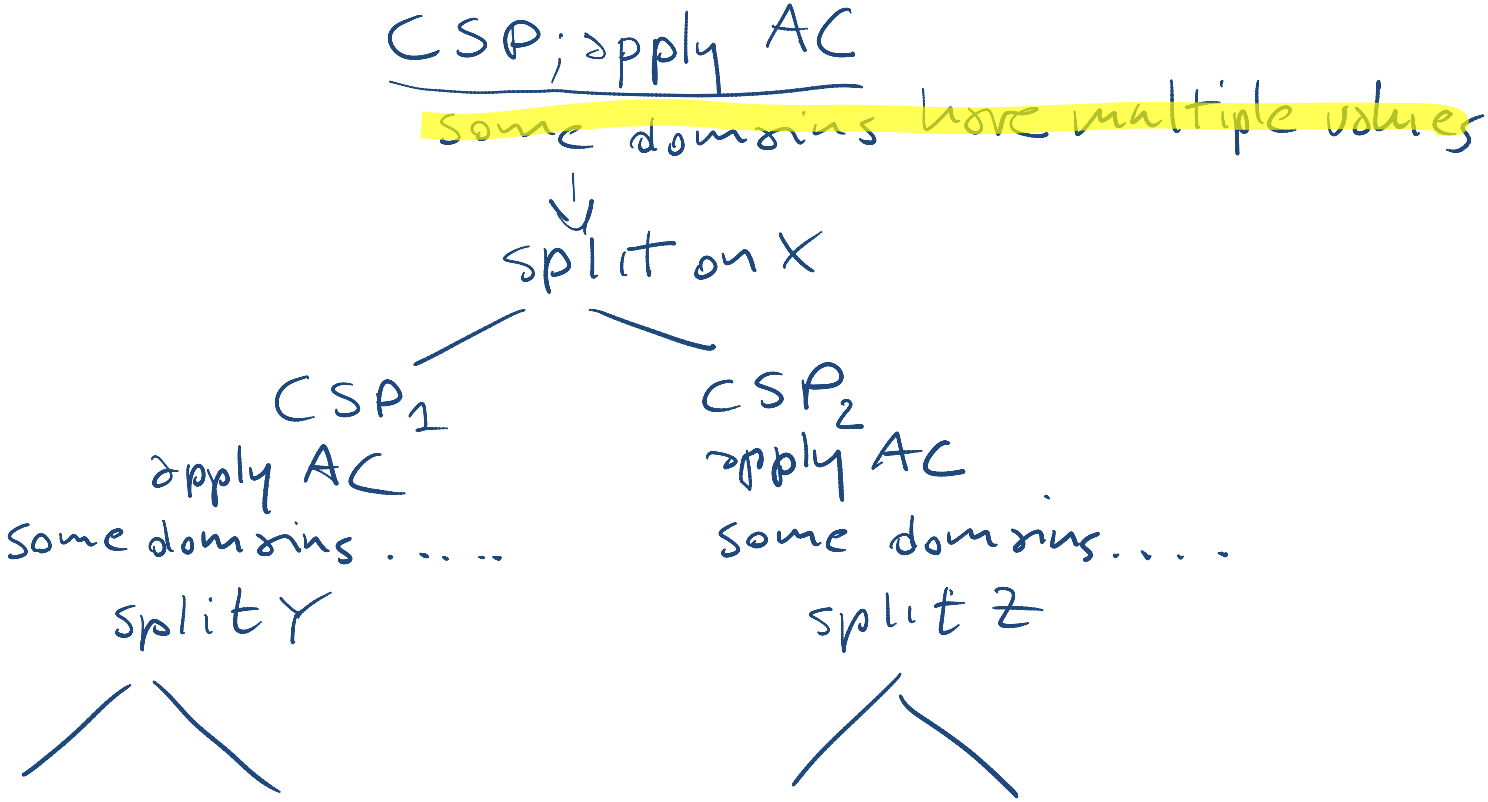
By reducing dom(X) we may be able to…

- 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



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

Searching by domain splitting

CSP, apply AC

If domains with multiple values

Split on one

CSP_1, apply AC

CSP_2, apply AC

If domains with multiple values
Split on one

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

O(2m) **O(2^m)** **O(m^2)** **O(2+m)**

Final set of 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

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**

For next class

Will be Posted on WebCT today

- Assignment2 (due next Thurs!\)

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

- **Read textbook:**
  - **4.8 we will finish CSPs**

- **MIDTERM: Mon May 28th – 3PM (room TBA)**

322 Feedback ☺ or ☹

- Lectures
- Slides
- Practice Exercises
- Assignments
- Alspace
- ……

- Textbook
- Course Topics / Objectives
- TAs
- Learning Goals
- ……

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**

Extra slide (may be used here?)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Goal state: 9×9 grid completely filled so that

- each column,
- each row, and
- each of the nine 3×3 boxes
- contains the digits from 1 to 9, only *one* time each

A possible **start state** (partially completed grid)

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Lecture Overview

- Recap of Lecture 14
- Domain Splitting for Arc Consistency
- Intro to Local Search (time permitting)

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

YES

NO

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

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

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

arcs <Z, r(Z,X)>

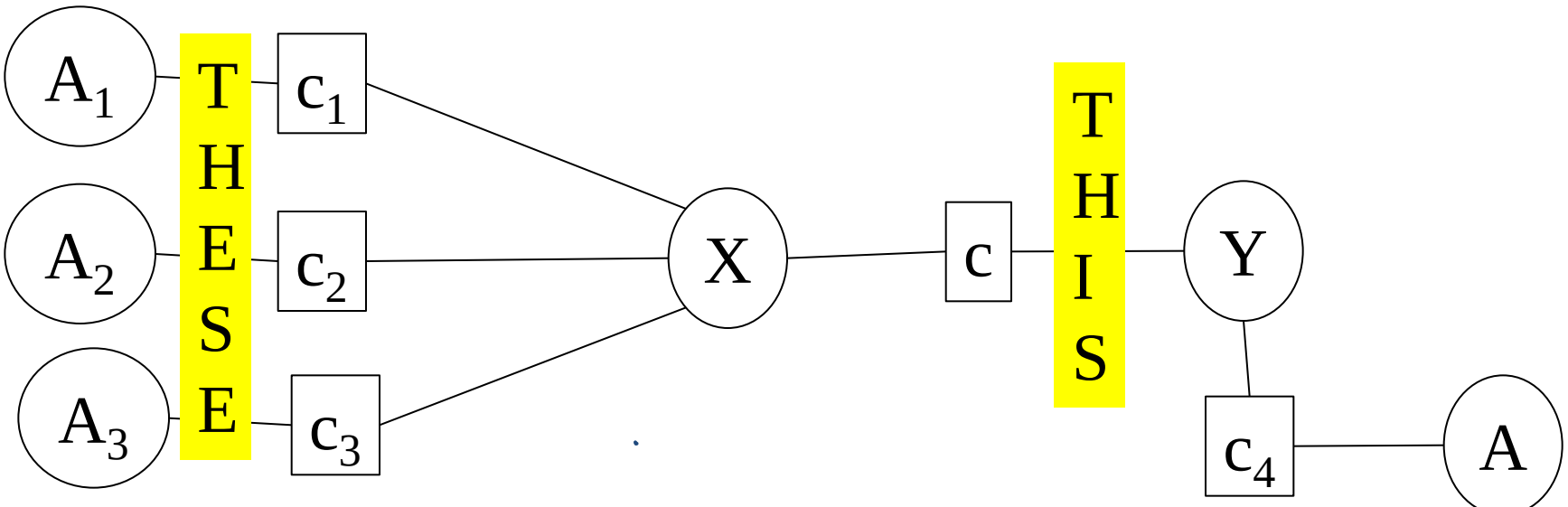
arcs <Z, r(Z,X)> and <X, r(Z,X)>

All arcs

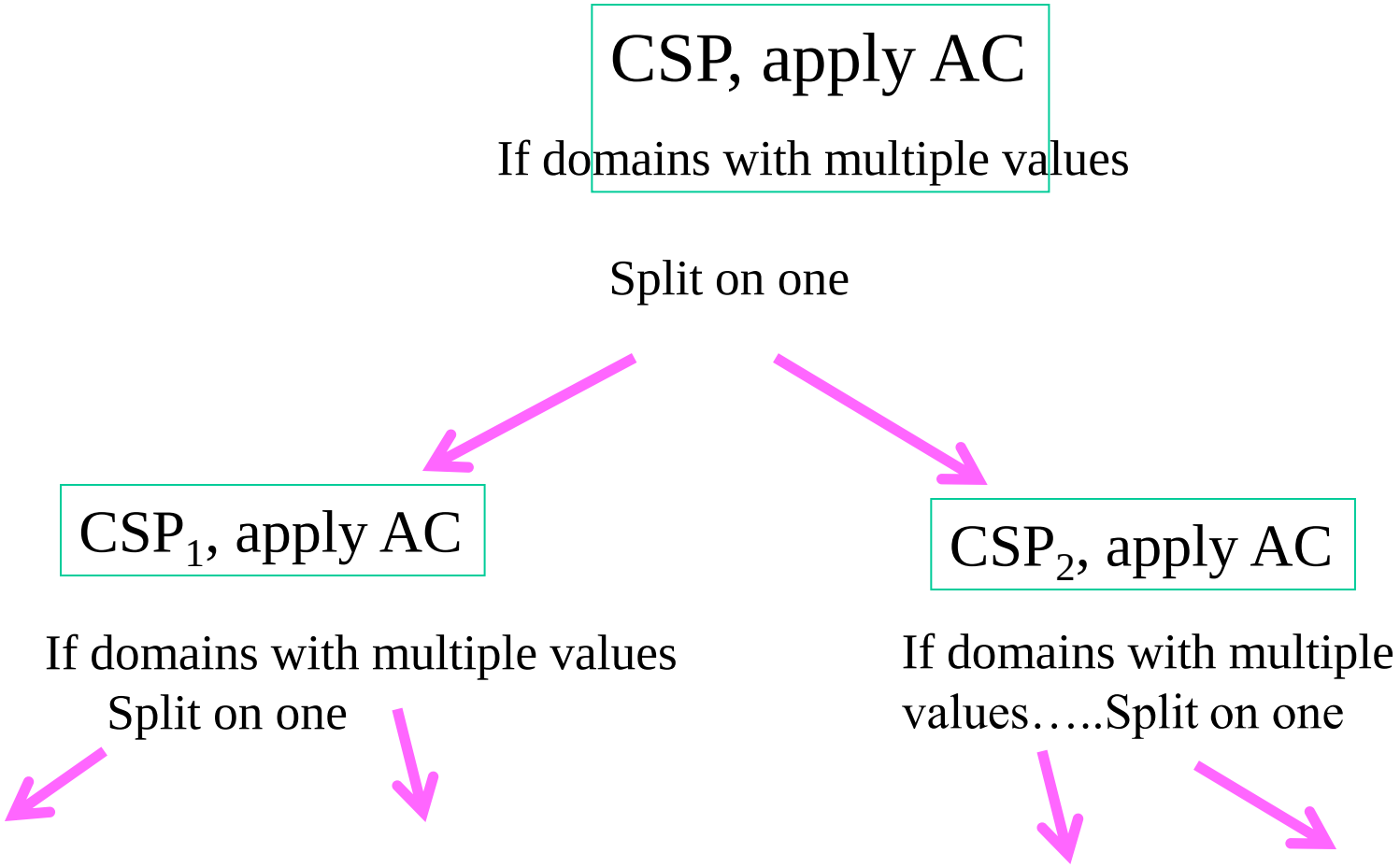
Domain splitting

- 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

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

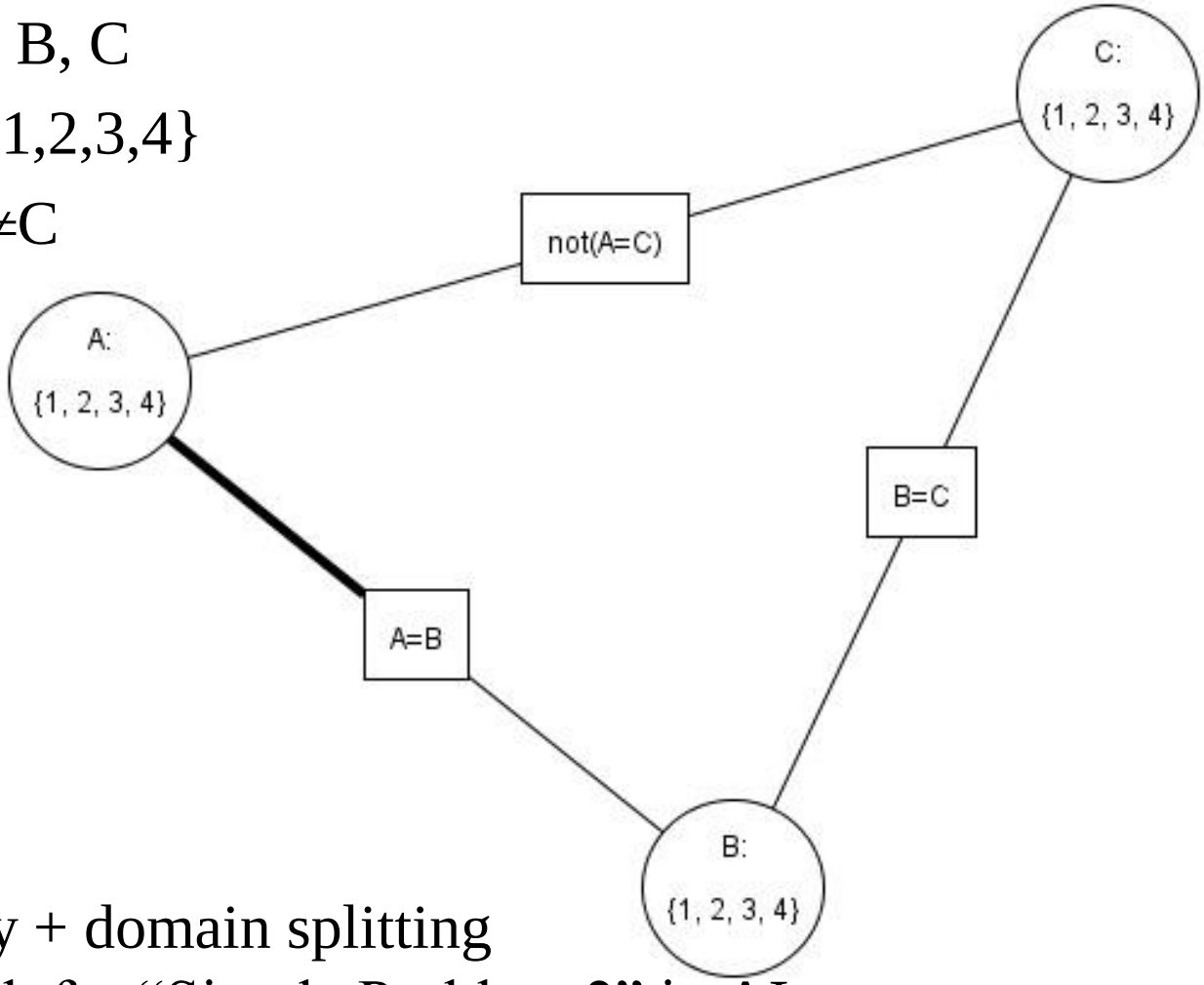


Searching by domain splitting



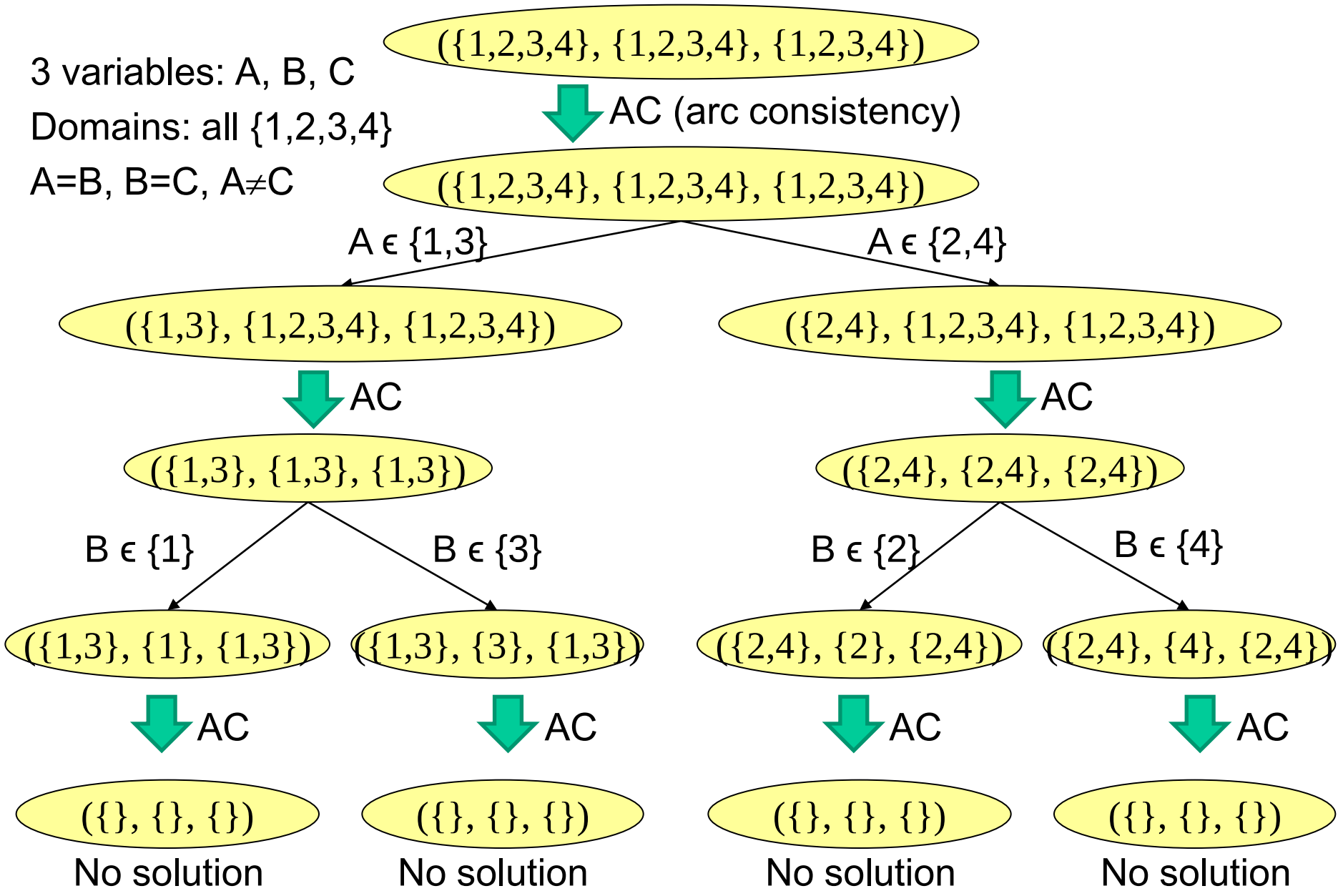
Domain Splitting in Action:

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

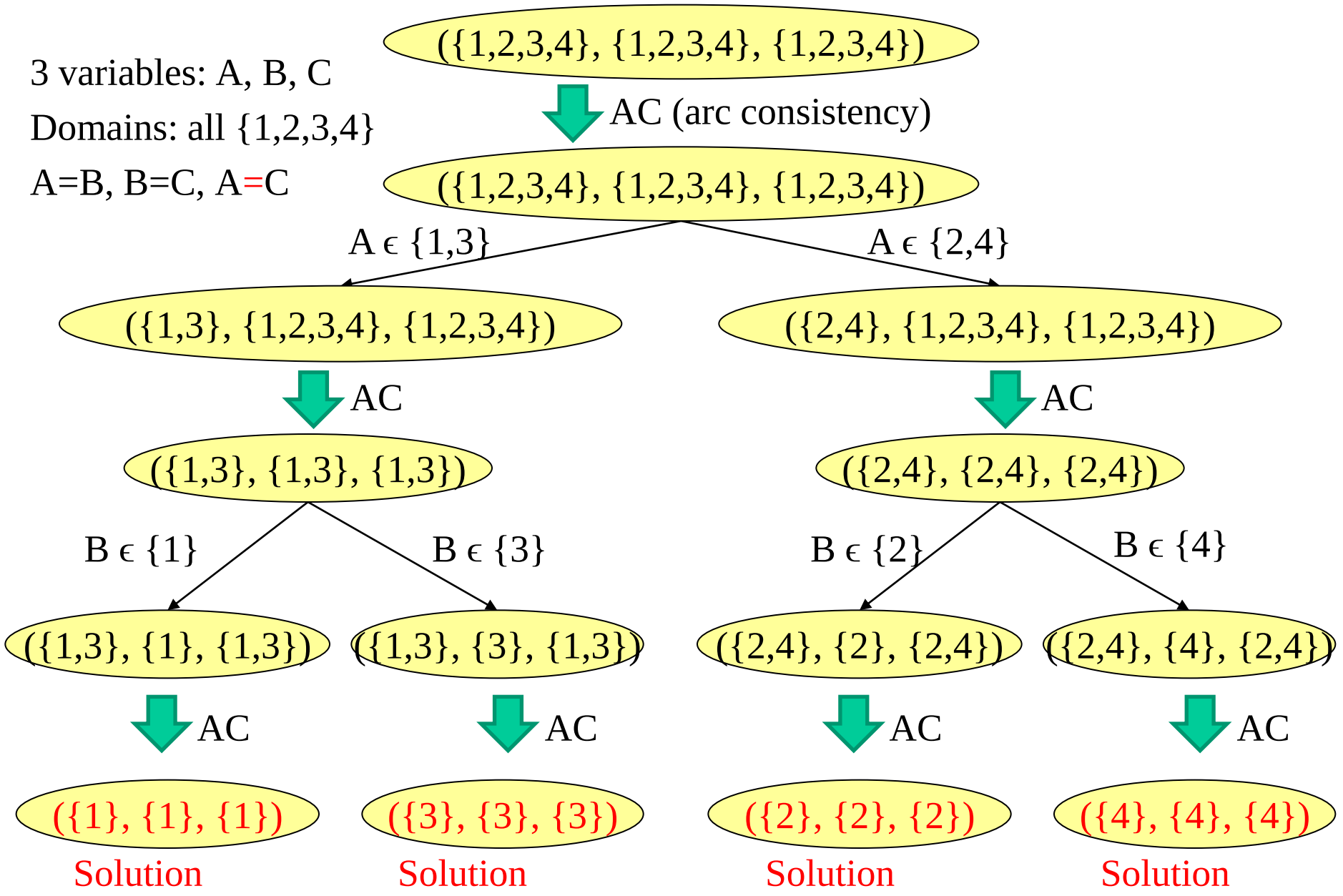


- Let's trace
  arc consistency + domain splitting
  for this network for "Simple Problem 2" in AIspace

Arc consistency + domain splitting: example



Arc consistency + domain splitting: another example



Another formulation of CSP as search

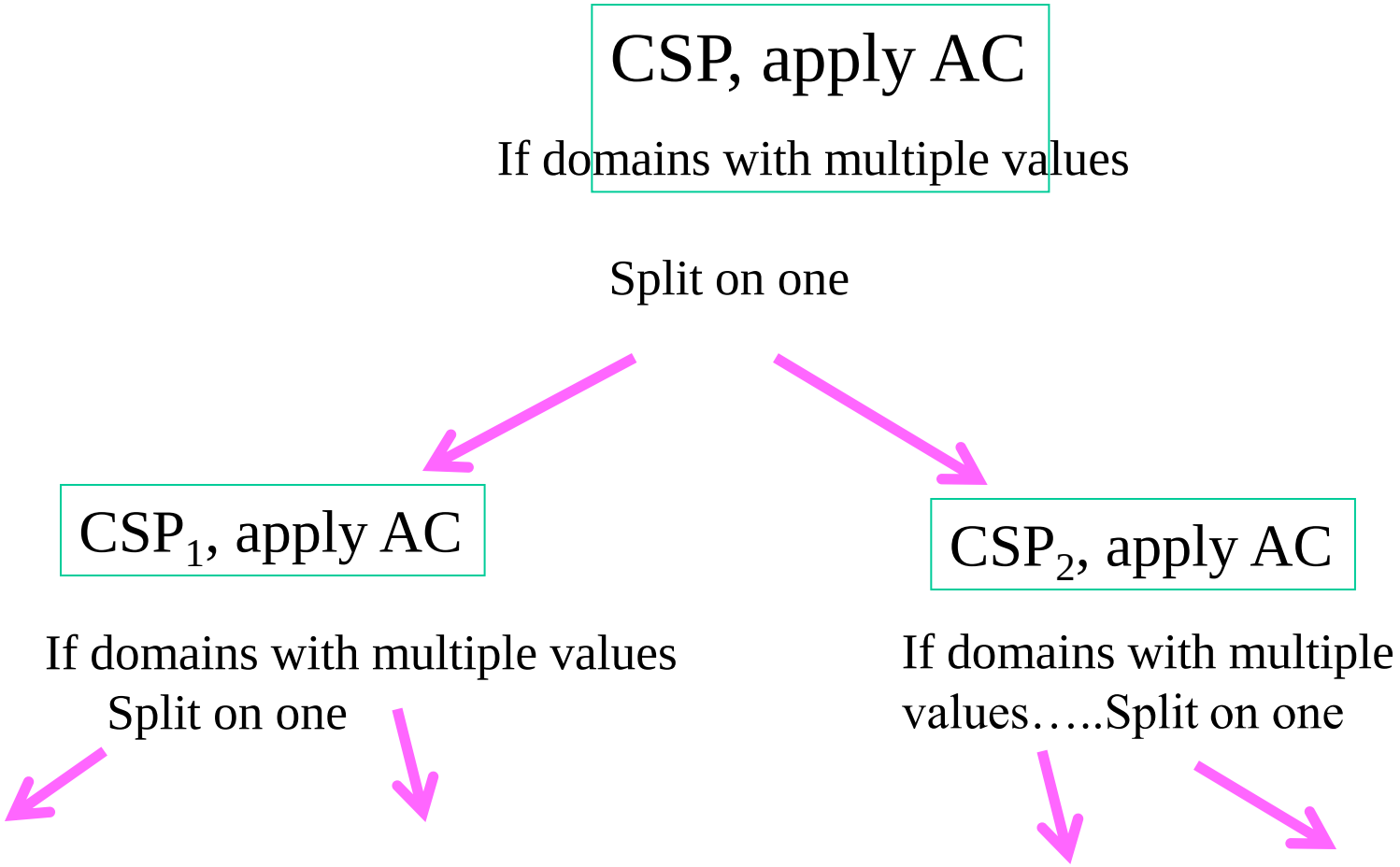
- Arc consistency with domain splitting
- State
  domains.
  With $D(V_i) \subseteq dom(V_i)$ for each V_i
- Start state: vector of original domains ($dom(V_1)$, …, $dom(V_n)$)
- Goal st tisfies all constraints
  - That is, only one value left for each variable
  - The assignment of each variable to its single value is a

Another formulation of CSP as search

- Arc consistency with domain splitting
- States: vector $(D(V_1), \ldots, D(V_n))$ of remaining domains,

  with $D(V_i) \subseteq dom(V_i)$ for each V_i
- Start state: vector of original domains $(dom(V_1), \ldots, dom(V_n))$
- 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

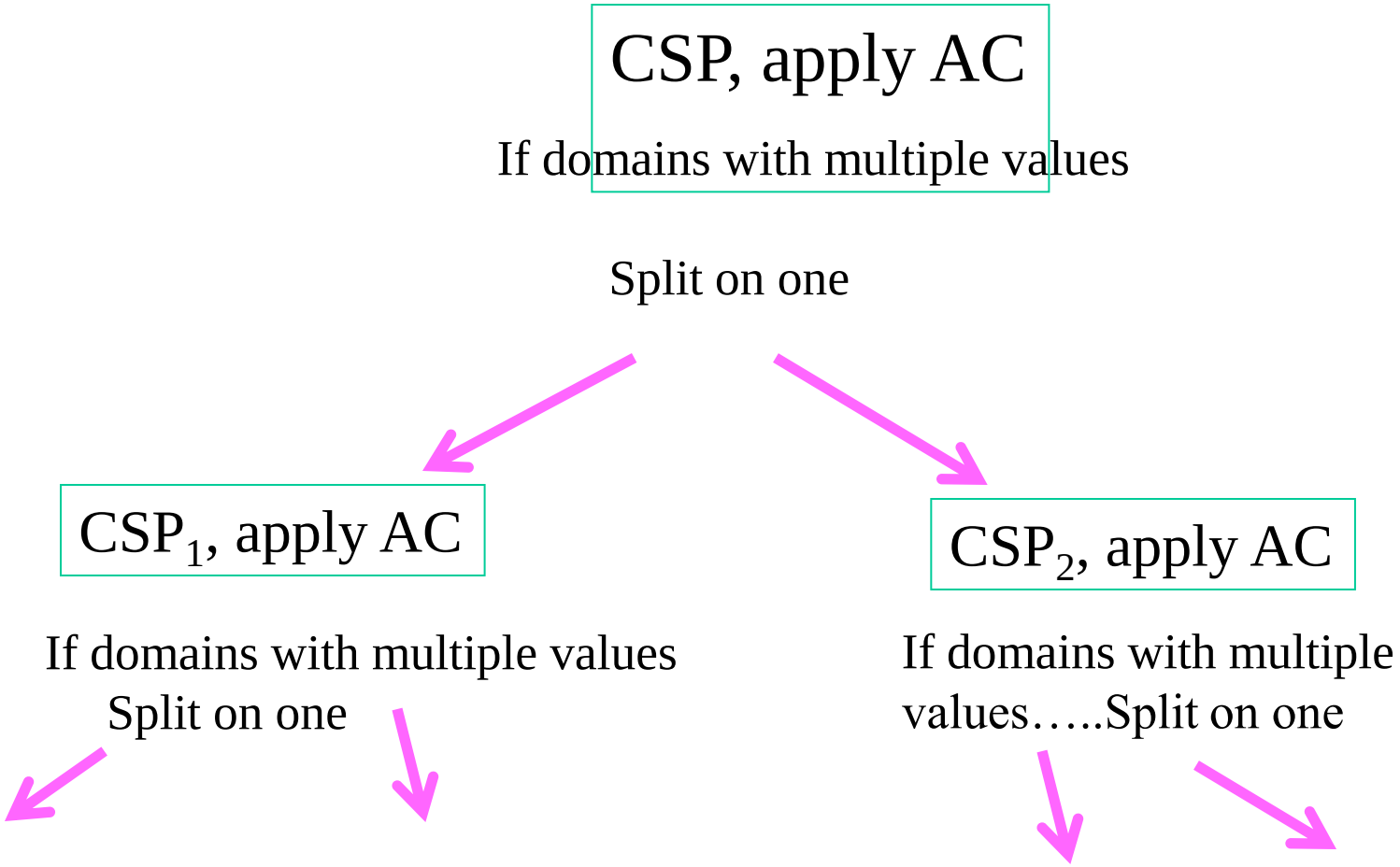
Searching by domain splitting



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

O(2m) **O(2^m)** **O(m^2)** **O(2+m)**

Searching by domain splitting



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

O(2m): It is DFS