Constraint Satisfaction Problems

Mostly adapted from chapter 4 of the AI textbook by
David Poole and Alan Macworth

Constraint Satisfaction Problem

An instance of CSP is characterised by:

- A set of **variables** $V_1, V_2, \ldots, V_n$
- A set of associated **domains**, one for each variable. A domain D_{V_i} is the set of values that V_i can assume.
- A set of **constraints** on various subsets of the variables which specify legal combinations of values for these variables.

A **total assignment** is an assignment of a value to every variable (i.e. an n-tuple from the cartesian product of the domains).

A **solution** to the CSP is a total assignment that satisfies all the constraints.

Example CSP

Variables: *A, B, C*

Domains: $D_A = \{-1, 0, 1\}$ $D_B = \{1,2,3,4\}$ $D_C = \{1, 2, 3, 4\}$

Constraints: $A < B$ and $A + B = C$

Some solutions:

A = 0, B = 3, C = 3

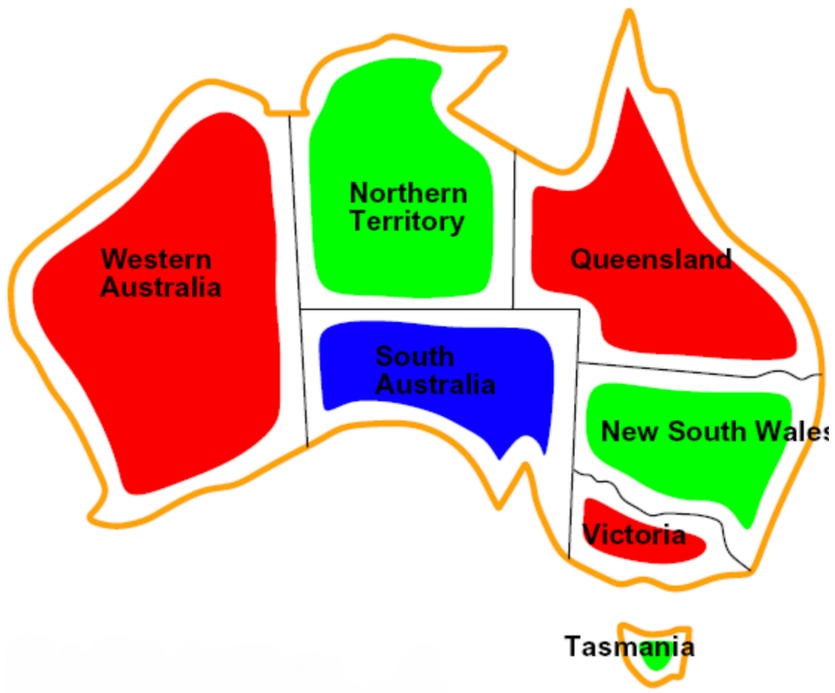
A = -1, B = 2, C = 1

Example CSP: Australian map colouring

- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: the same for all variables: {red, green, blue}
- Constraints: adjacent regions must have different colours. (e.g. WA≠NT, ...)

Solutions are assignments satisfying all constraints, e.g.:

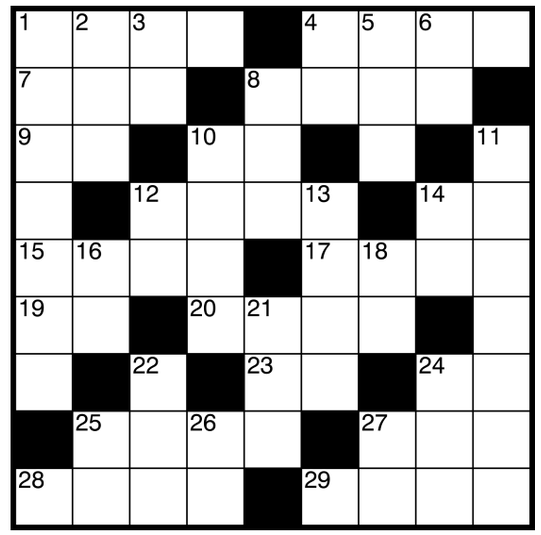
$$\{WA = red, NT = green, Q = red,$$
$$NSW = green, V = red, SA = blue, T = green\}$$

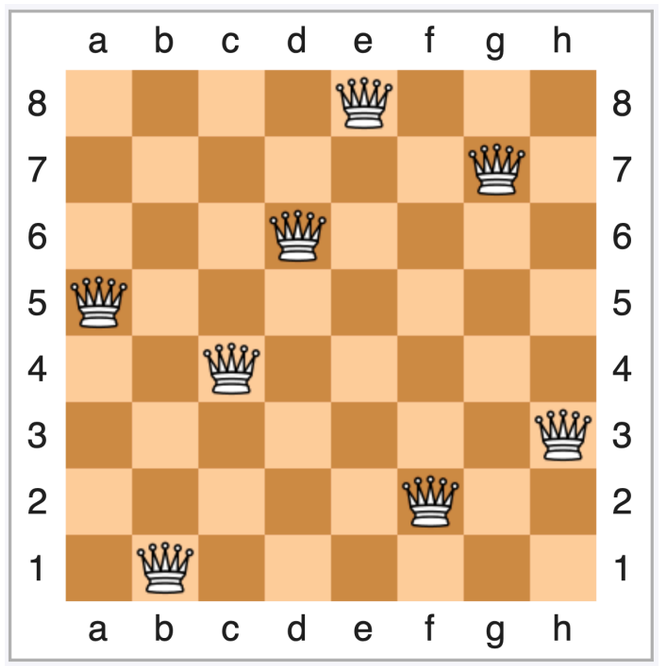


More CSP examples: puzzles

- Sudoko
- Crosswords
- Eight queens puzzle

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





Industrial CSP examples

- Scheduling, Timetabling, Rostering

- Car sequencing (assembly lines)

- Positioning agents with some constraints. For example:
  - distance between each pair greater than a threshold
  - distance of the closest agent less than a threshold
  - distance from a command centre less than a threshold

CSP algorithms

- CSP is difficult (why?) but
  - it has many applications;
  - some instances can be solved more efficiently.
- We will look at several algorithms:
  - Generic algorithms:
    - Generate and test (exhaustive search)
    - Backtracking
    - Using graph search
  - More specialised algorithms:
    - Arc consistency
    - Domain splitting
    - Variable elimination

Generate-and-Test Algorithm

- Generate the assignment space $\mathbf{D} = \mathbf{D}_{V_1} \times \mathbf{D}_{V_2} \times \ldots \times \mathbf{D}_{V_n}$. Test each assignment with the constraints.
- Example:

$$\begin{aligned} \mathbf{D} &= \mathbf{D}_A \times \mathbf{D}_B \times \mathbf{D}_C \times \mathbf{D}_D \times \mathbf{D}_E \\ &= \{1,2,3,4\} \times \{1,2,3,4\} \times \{1,2,3,4\} \\ &\quad \times \{1,2,3,4\} \times \{1,2,3,4\} \\ &= \{\langle 1,1,1,1,1\rangle, \langle 1,1,1,1,2\rangle, \ldots, \langle 4,4,4,4,4\rangle\}. \end{aligned}$$

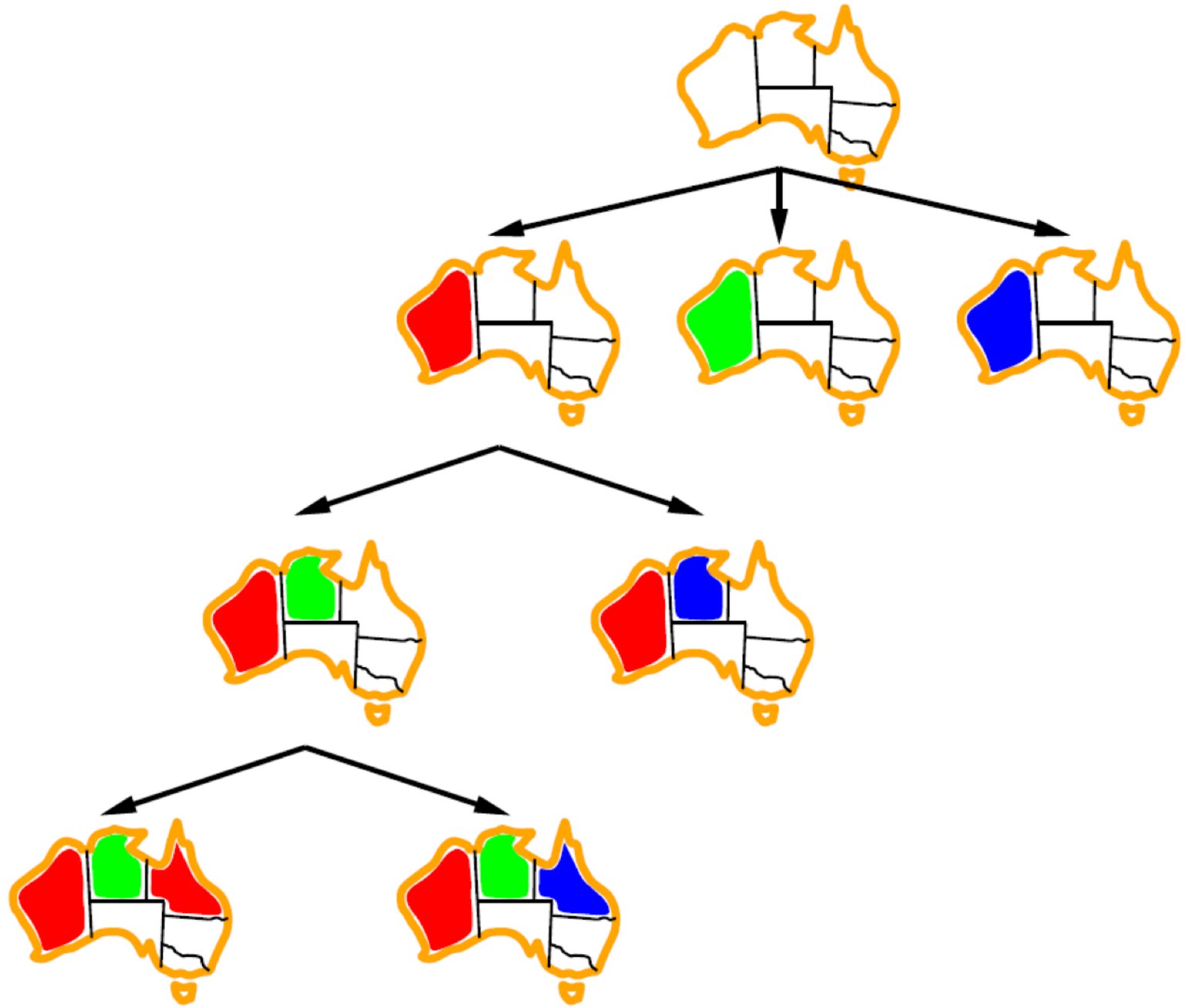
- Generate-and-test is always exponential in the number of variables.

Backtracking algorithm

- Systematically explore **D** by instantiating the variables one at a time
- evaluate each constraint predicate as soon as all its variables are bound
  - any partial assignment that doesn't satisfy the constraint can be pruned.

Example Assignment $A = 1 \wedge B = 1$ is inconsistent with constraint $A \neq B$ regardless of the value of the other variables.

Backtracking example



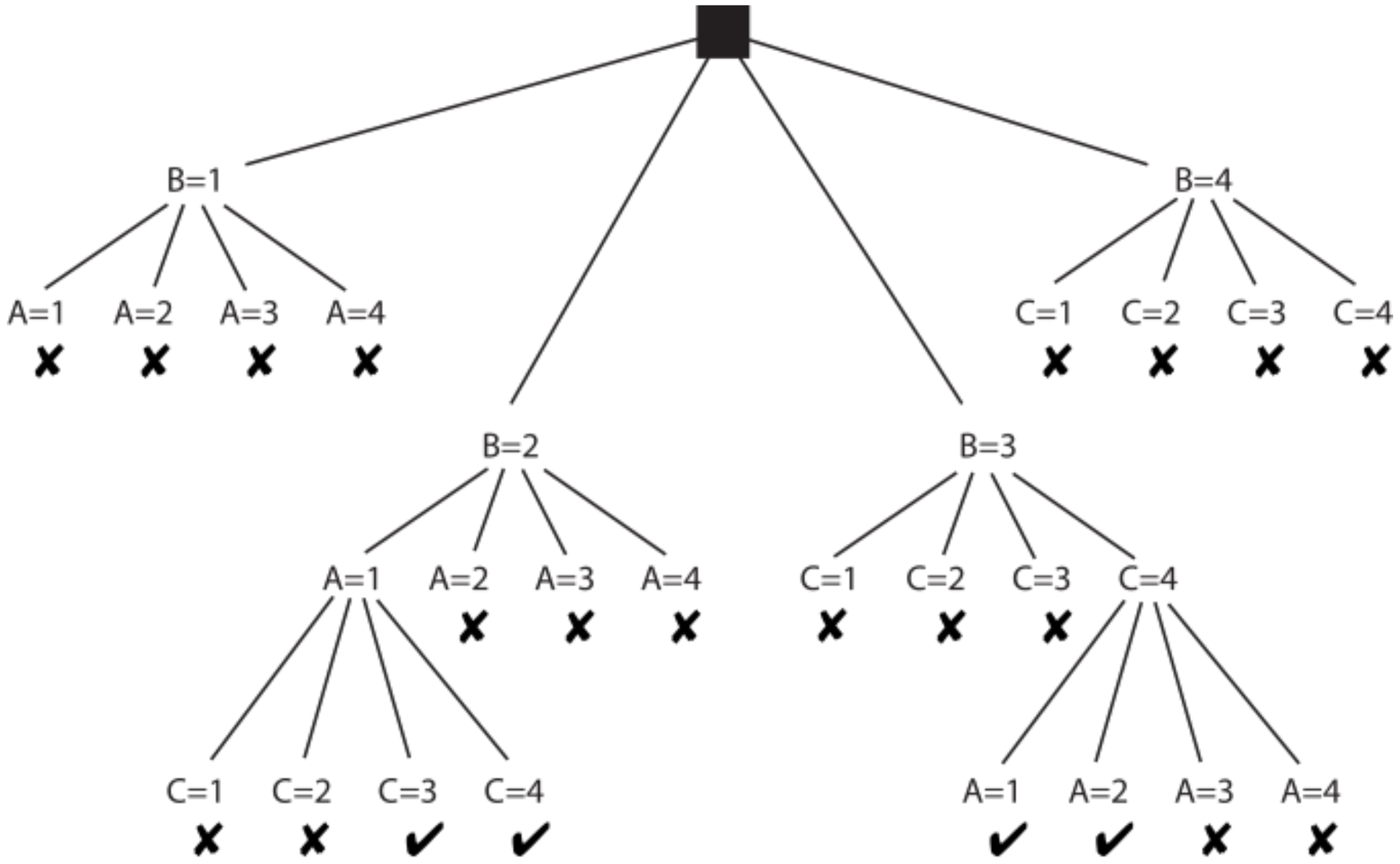
CSP as graph search

A CSP can be represented as a graph-searching algorithm:

- A node is an assignment values to some of the variables.
- Suppose node N is the assignment $X_1 = v_1, \ldots, X_k = v_k$.
  Select a variable Y that isn't assigned in N.
  For each value $y_i \in dom(Y)$ there is a neighbour $X_1 = v_1, \ldots, X_k = v_k, Y = y_i$ if this assignment is consistent with the constraints on these variables.
- The start node is the empty assignment.
- A goal node is a total assignment that satisfies the constraints.

Graph search example

- Consider a CSP with variables A, B, and C each with domain {1, 2, 3, 4}. The constraints are A < B and B < C. The search tree may look like this:



Constraint networks

An instance of CSP can be represented as a network (special graph):

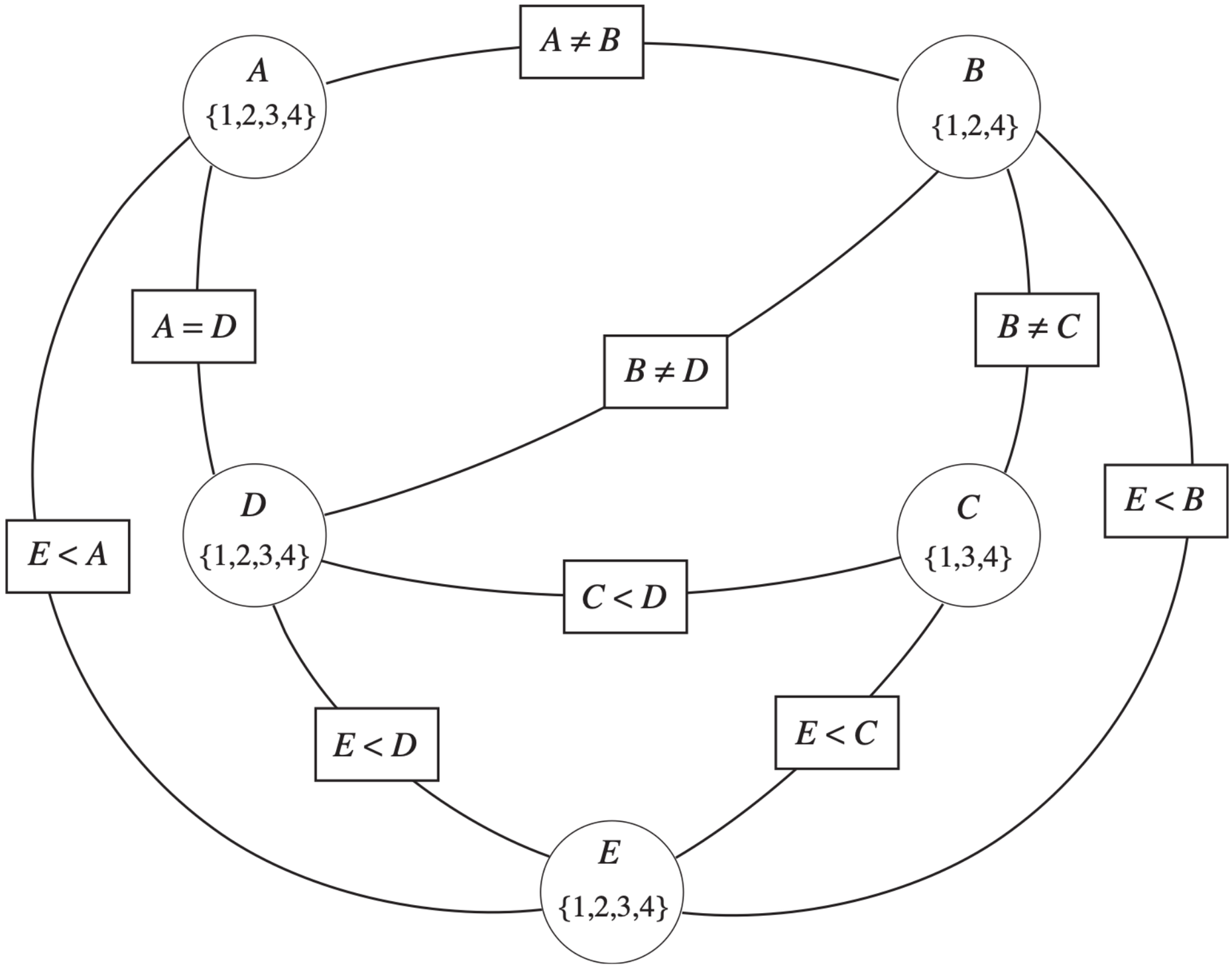
Nodes:

- One for each variable and its domain (usually drawn as an oval)
- One for each constraint (usually drawn as a rectangle)

Arcs:

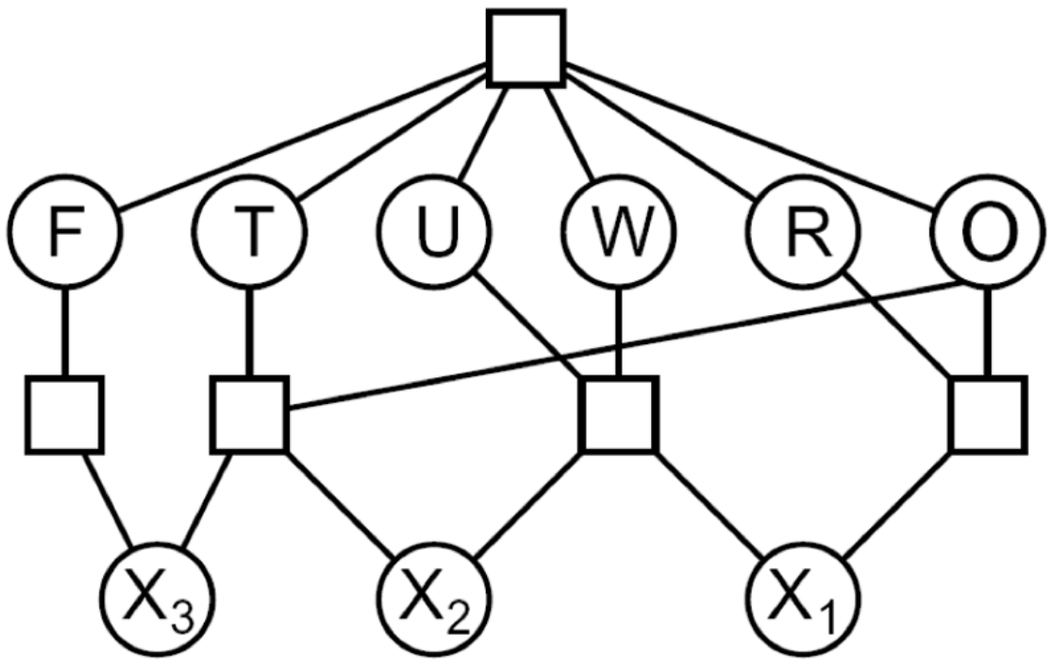
- There is an arc from variable X to each constraint that involves X.

Example: Constraint Network



Example CSP: Cryptarithmetic

```
    T W O
+   T W O
-----------
F O U R
```



Variables:

T W F O U R X1 X2 X3

Domains of T, W, F, O, U, R:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Domains of X1, X2, X3:

0, 1

Constraints:

*O + O = R + 10*X1*

*W + W + X1 = U + 10*X2*

…

alldiff(T, W, F, O, U R)

Arc consistency

Let's represent a constraint r over variables X, Y_1, Y_2, …, Y_n as r(X, Y_1, Y_2, …, Y_n) or simply $r(X, \overline{Y})$

- An arc $\langle X, r(X, \overline{Y}) \rangle$ is arc consistent if, for each value $x \in dom(X)$, there is some value $\overline{y} \in dom(\overline{Y})$ such that $r(x, \overline{y})$ is satisfied.
- A network is arc consistent if all its arcs are arc consistent.
- If an arc $\langle X, r(X, \overline{Y}) \rangle$ is *not* arc consistent, all values of X in $dom(X)$ for which there is no corresponding value in $dom(\overline{Y})$ may be deleted from $dom(X)$ to make the arc $\langle X, r(X, \overline{Y}) \rangle$ consistent.

Arc consistency example

Consider an instance of CPS with variables A, B, C where the domains of all the variables are {1, 2, 3, 4} and the only constraint is A + B = C. Make the arc <A, A+B=C> arc-consistent (one iteration only).

General arc consistency algorithm

- Takes a constraint network and makes all the arcs arc-consistent.
- Iterates through arcs making each consistent.
- An arc $< X, r(X, \bar{Y}) >$ needs to be revisited if the domain of one of the Y's is reduced.

General Arc Consistency (GAC) Algorithm

for each variable X:
    $D_X := dom(X)$
$to\_do := \{\langle X, c\rangle \mid c \in C$ and $X \in scope(c)\}$
while to_do is not empty:
    select and **remove** path $\langle X, c\rangle$ from to_do
    suppose scope of c is $\{X, Y_1, \ldots, Y_k\}$
    $ND_X := \{x \mid x \in D_X$ and
                exists $y_1 \in D_{Y_1}, \ldots, y_k \in D_{Y_k}$
                s.th. $c(X = x, Y_1 = y_1, \ldots, Y_k = y_k) = true\ \}$
    if $ND_X \neq D_X$:
        $to\_do := to\_do \cup \{\langle Z, c'\rangle \mid X \in scope(c'),$
                c' is not c, $Z \in scope(c') \setminus \{X\}\}$
        $D_X := ND_X$
return $\{D_X \mid X$ is a variable$\}$

Example: General Arc Consistency Algorithm

Use the General Arc Consistency algorithm to make the following instance of CSP arc-consistent.

There are 3 variables: A, B, and C with domains {1, 2, 3, 4}. There are two constraints: A < B and B < C.

General arc consistency output

There are three possible outcomes (when all arcs are consistent):

- One or more domains become empty: no solution

- Each Domain has exactly a single value: a unique solution

- Some domains have more than one value (and none are empty): there may or may not be a solution [further search is required]

If further search needs to be done, we can perform domain splitting.The idea is to split a problem into a number of disjoint cases and solve each case separately. the set of all solutions to the initial problem is the union of the solutions to each case:

- pick a variable whose domain has more than one value;

- split the domain (ideally in half);

- we only need to revisit arcs affected by the split;

- recursively solve the resulting instances of CSP and return the union of the solutions.

CSP Solver using Arc Consistency and Domain Splitting algorithms

Solve_all(*CSP*, *domains*) :
    simplify *CSP* with arc-consistency
    if one domain is empty:
        return $\{\}$
    else if all domains have one element:
        return {solution of that element for each variable}
else:
    select variable X with domain D and $|D| > 1$
    partition D into D_1 and D_2
    return *Solve_all*(*CSP*, *domains* with $dom(X) = D_1) \cup$
        Solve_all(*CSP*, *domains* with $dom(X) = D_2$)

Arc Consistency and Domain Splitting as Search

Domain splitting leads to search space

- Nodes: CSP with arc-consistent domains
- Neighbors of *CSP*:
  if all domains are non-empty:
  select variable X with domain D and $|D| > 1$
  partition D into D_1 and D_2
  neighbors are
  - $make\_AC(CSP \mid dom(X) = D_1)$
  - $make\_AC(CSP \mid dom(X) = D_2)$
- Goal: all domains have size 1
- Start node: $make\_AC(CSP)$

Variable Elimination

Idea: eliminate the variables one-by-one passing their constraints to their neighbours.

- Consider a CSP that contains a number of variables A, B, C, … each with domain {1, 2, 3, 4}.
- Suppose the constraints that involve B are A < B and B < C.
- There may be many other variables, but if B does not have any constraints in common with these variables, eliminating B will not impose any new constraints on these other variables.
- To remove B, first join on the relations that involve B.
- To get the relation on A and C induced by B, project this join onto A and C.

A	B
1	2
1	3
1	4
2	3
2	4
3	4

$\bowtie$

B	C
1	2
1	3
1	4
2	3
2	4
3	4

$=$

A	B	C
1	2	3
1	2	4
1	3	4
2	3	4

A	C
1	3
1	4
2	4

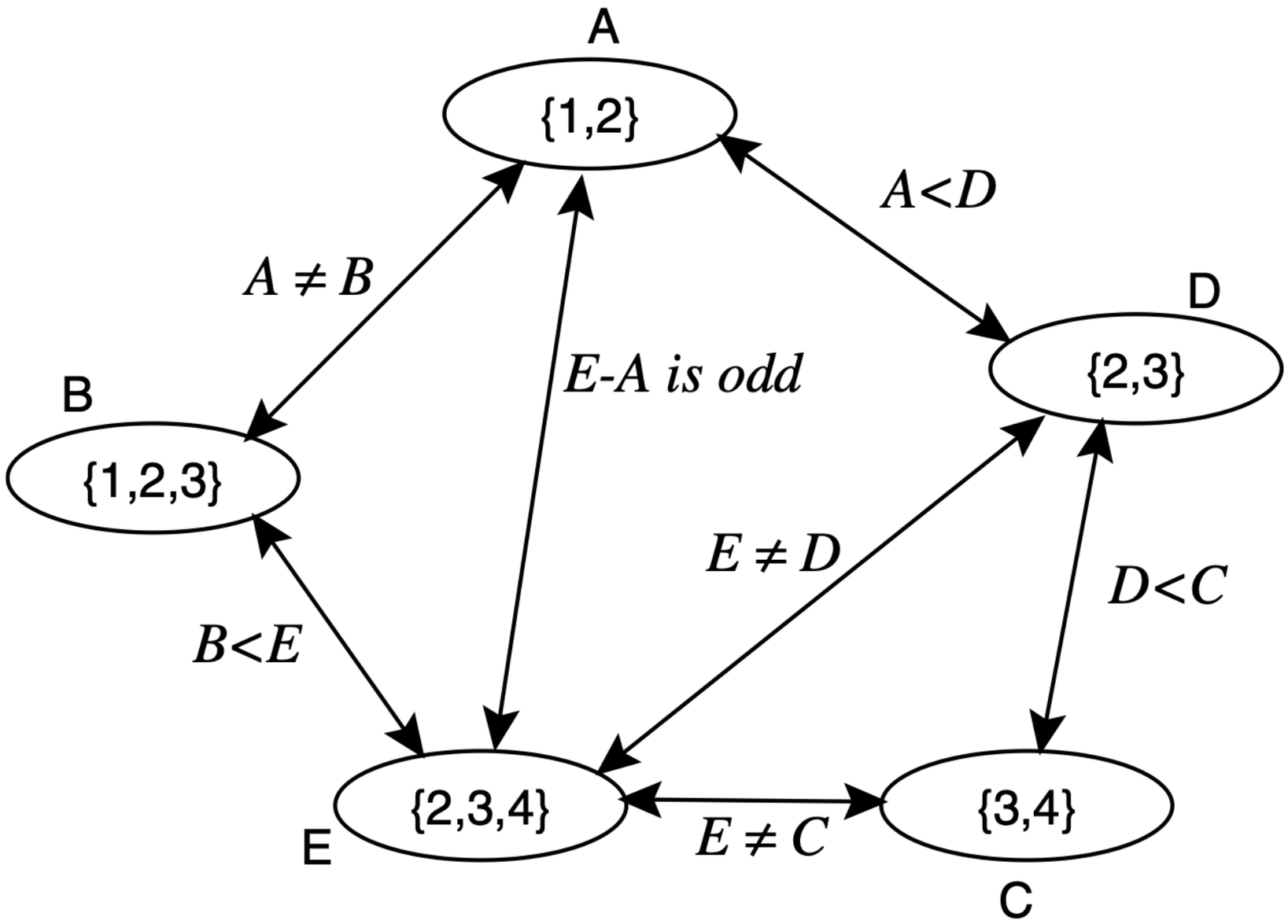
Variable Elimination Algorithm (the outline)

- If there is only one variable, return the intersection of the (unary) constraints that contain it
- Select a variable X
- Join the constraints in which X appears, forming constraint R_1
- Project R_1 onto its variables other than X, forming R_2
- Replace all of the constraints in which X appears by R_2
- Recursively solve the simplified problem, forming R_3
- Return R_1 joined with R_3

Variable Elimination Algorithm

1: **procedure** $VE\_CSP(Vs, Cs)$
2: **Inputs**
3: Vs: a set of variables
4: Cs: a set of constraints on Vs
5: **Output**
6: a relation containing all of the consistent variable assignments
7: **if** Vs contains just one element **then**
8: return the join of all the relations in Cs
9: **else**
10: **select** variable X to eliminate
11: $Vs' := Vs \setminus \{X\}$
12: $Cs_X := \{T \in Cs : T \text{ involves } X\}$
13: **let** R be the join of all of the constraints in Cs_X
14: **let** R' be R projected onto the variables other than X
15: $S := VE\_CSP(Vs', (Cs \setminus Cs_X) \cup \{R'\})$
16: **return** $R \bowtie S$

Variable Elimination Example



Variable Elimination Example (cont.)

$r_1 : C \neq E$	C	E
	3	2
	3	4
	4	2
	4	3

$r_2 : C > D$	C	D
	3	2
	4	2
	4	3

$r_3 : r_1 \bowtie r2$	C	D	E
	3	2	2
	3	2	4
	4	2	2
	4	2	3
	4	3	2
	4	3	3

$r_4 : \pi_{\{D,E\}} r_3$	D	E
	2	2
	2	3
	2	4
	3	2
	3	3

$\hookrightarrow$ new constraint

Variable Elimination Example (cont.)