CONSTRAINT SATISFACTION PROBLEMS (CSP)

DEFINITION OF A CSP

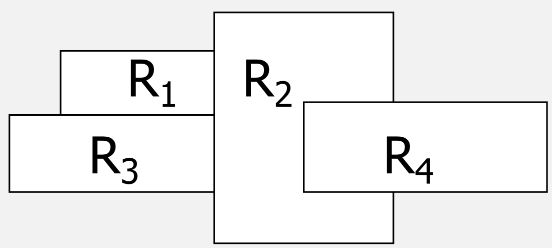
- **V** (a set of variables) with domains $\mathbf{D_i}$ not empty of possible values.
- **C** (a set of restrictions) where are involved a set of variables.
- The goal: To find an assignation of values for all the variables (complete) and they satisfy the set of restrictions (consistent).
- Some CSPs also require a solution that maximize an objective function (un such case we have an optimization problem).

TYPES OF CSP PROBLEMS

- Computational Vision
- Job scheduling
- Circuits (chips) design
- Products design
- Sites configuration
- Etc. …

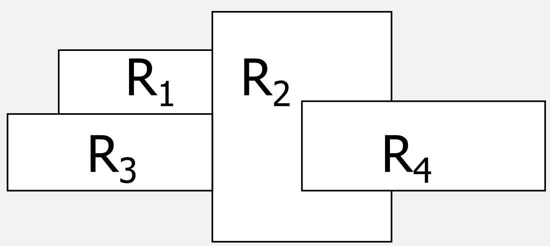
EXAMPLE: MAP COLORING

- Goal: To color a map of regions in such a way that no neighboring regions have the same color!.



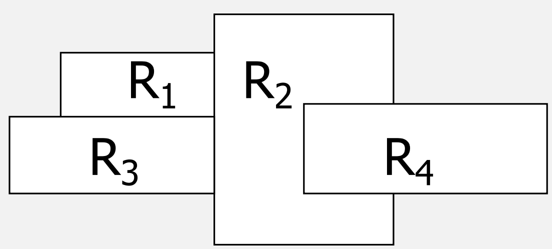
DEFINITION OF THIS PROBLEM

- **Variables**: regions = $\{R_1,R_2,R_3,R_4\}$
- **Domains**: possible colors: ({Blue,Green,Yellow})
- **Restrictions**: $R_1 \neq R_2$, $R_1 \neq R_3$, $R_2 \neq R_3$, $R_2 \neq R_4$

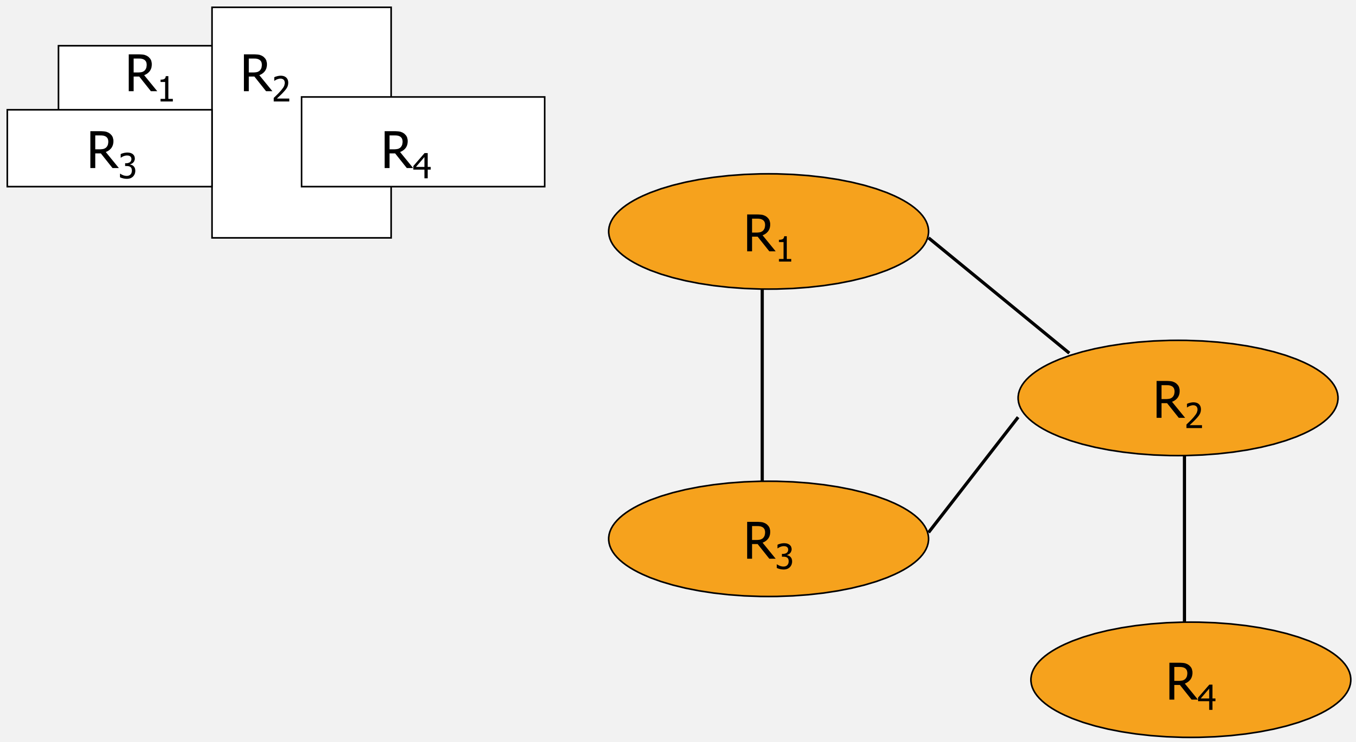


FORMAL DEFINITION OF RESTRICTIONS

- $R_1 \neq R_2$
- < (R1, R2), {(Blue,Green), (Blue,Yellow), (Green, Blue), (Green,Yellow), (Yellow, Blue), (Yellow, Green)}>



CONSTRAINT GRAPH



TYPES OF ENVIRONMENTS

- **Discrete and finite domains**
- Discrete and infinite domains
- Continuous domains

TYPES OF RESTRICTIONS

- By arity: unitarias, binarias, n-arias
- **Absolute constraints**: where we can not violate this constraints.
- **Preference constraints**: where the solution is prefered.

JOB SCHEDULING

- Variables: A, B, C, D, E: Meaning the start times of different machines in a factory.
- Domains : DA = {1,2,3,4}, DB = {1,2,3,4}, DC = {1,2,3,4}, DD = {1,2,3,4}, DE = {1,2,3,4}
- Restrictions:
- $(B \neq 3) \wedge (C \neq 2) \wedge (A \neq B) \wedge (B \neq C) \wedge$
  $(C < D) \wedge (A = D) \wedge (E < A) \wedge (E < B) \wedge (E < C) \wedge (E < D) \wedge (B \neq D)$.

THE MOST SIMPLE ALGORITHM: GENERATE & TEST

- $D = DA \times DB \times DC \times DD \times DE$
  $= \{1,2,3,4\} \times \{1,2,3,4\} \times \{1,2,3,4\} \times \{1,2,3,4\} \times \{1,2,3,4\}$
  $= \{\langle 1,1,1,1,1 \rangle, \langle 1,1,1,1,2 \rangle, \ldots, \langle 4,4,4,4,4 \rangle\}.$

- **Disadvantage: Exponential time!**

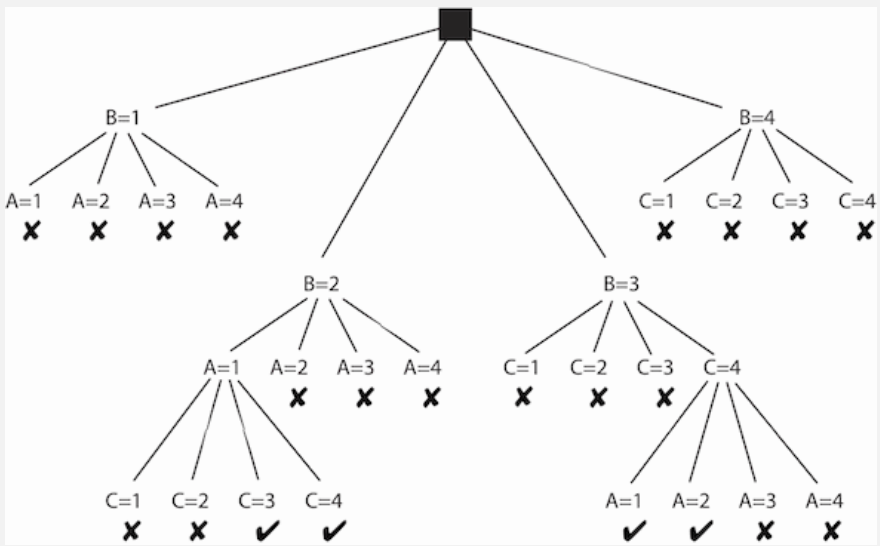
- **It is better to take a Search approach!**

PROBLEM FORMULATION FOR A CSP

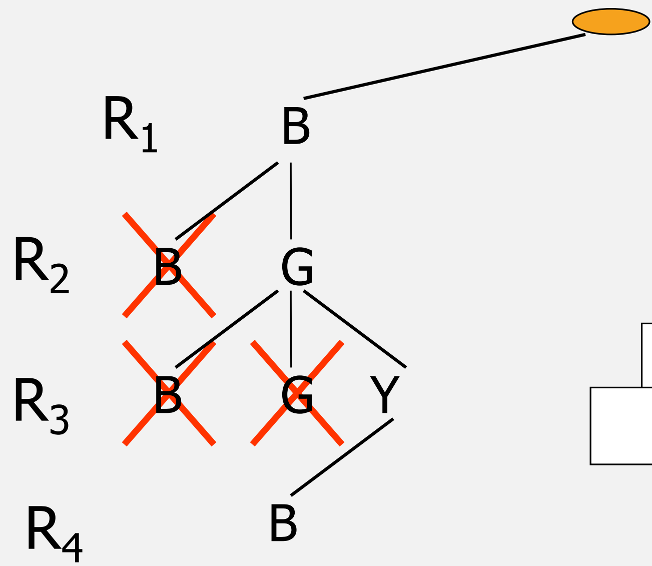
- **Initial State**: empty vector of vector assignations.
- **Succesor Function:** To assign values to variables that have been not assigned avoiding conflicts.
- **Goal State**: Complete vector of variables assigned with values.
- **Step Cost**: 1

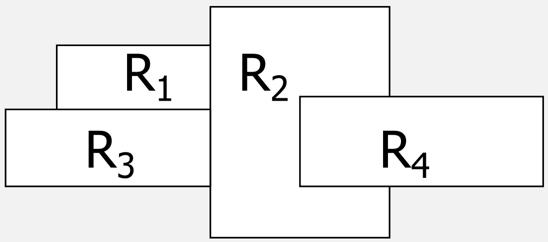
GRAPH SEARCHING ALGORITHM

- A similar example of the factory but with only three variables and two restrictions:
- Variables A, B y C. Domains: {1,2,3,4}. Restrictions: A<B and B<C.

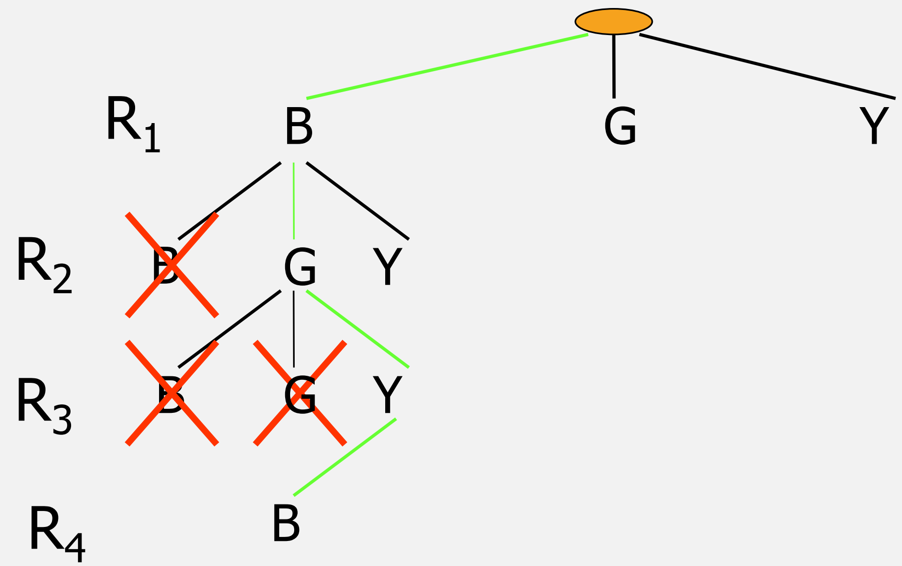


DEPTH-FIRST SEARCH WITH BACKTRACKING

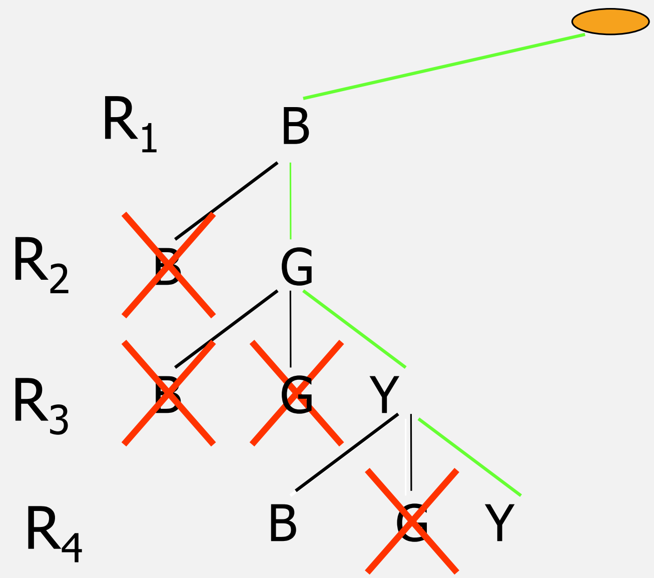




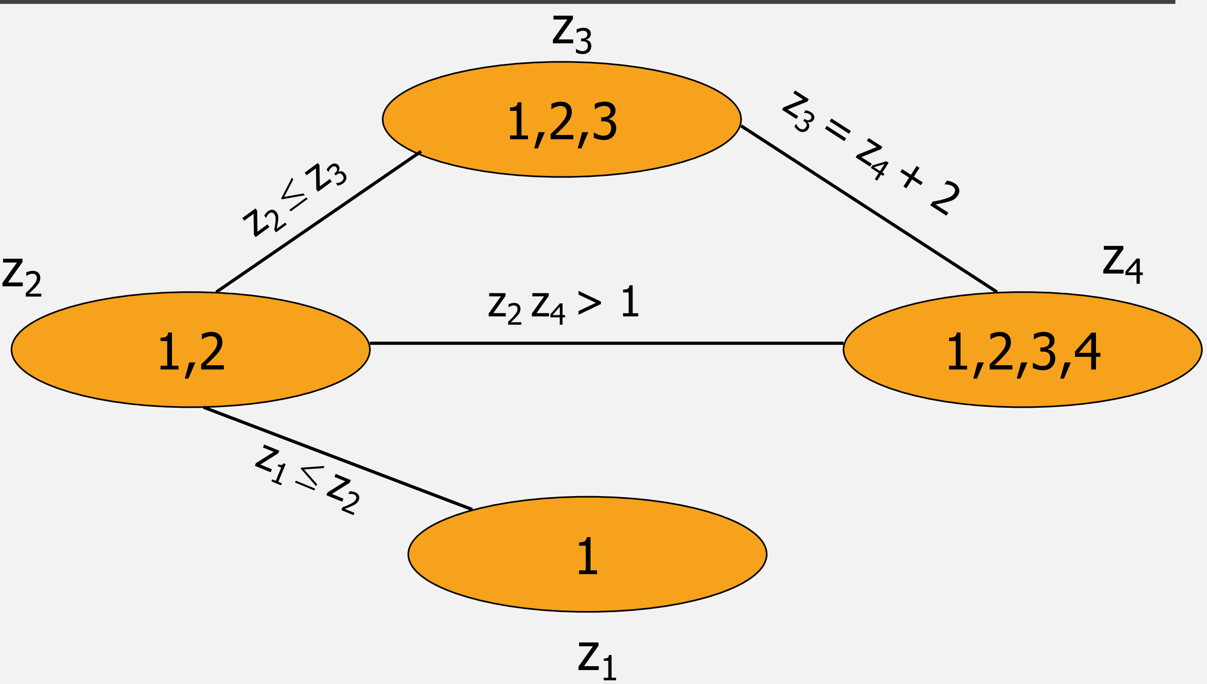
SOLUTION



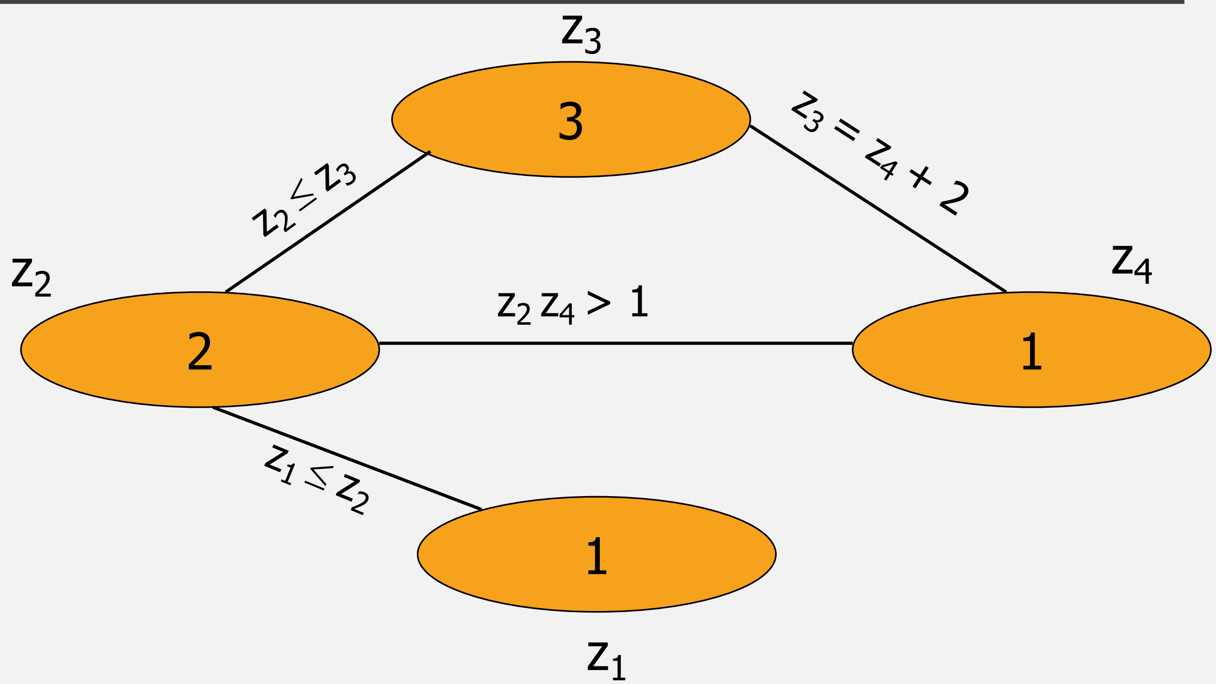
ANOTHER SOLUTION



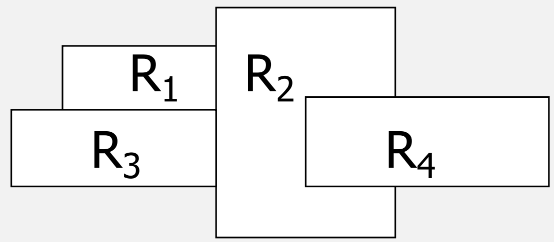
ANOTHER EXAMPLE OF A CSP



AFTER RUNNING AC-3



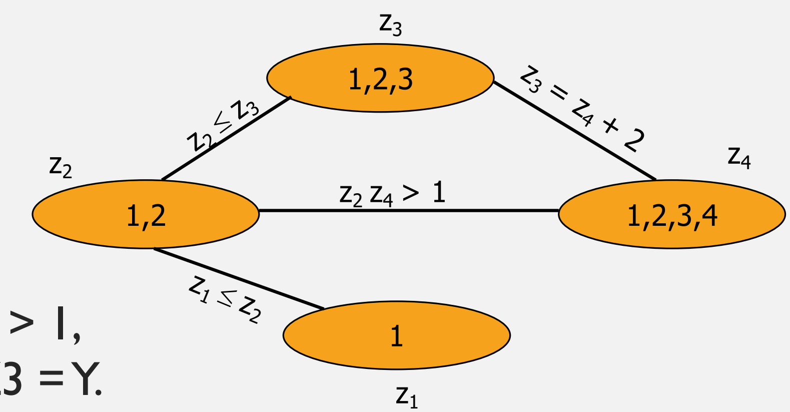
SOLVING THE COLORING MAP USING THE PROLOG LANGUAGE




```
dom(blue).
dom(green).
dom(yellow).

solution(R1,R2,R3,R4) :-
  dom(R1),
  dom(R2), R1 \== R2,
  dom(R3), R1 \== R3, R2 \== R3,
  dom(R4), R2 \== R4.
```

SOLVING THE SECOND EXAMPLE USING THE PROLOG LANGUAGE

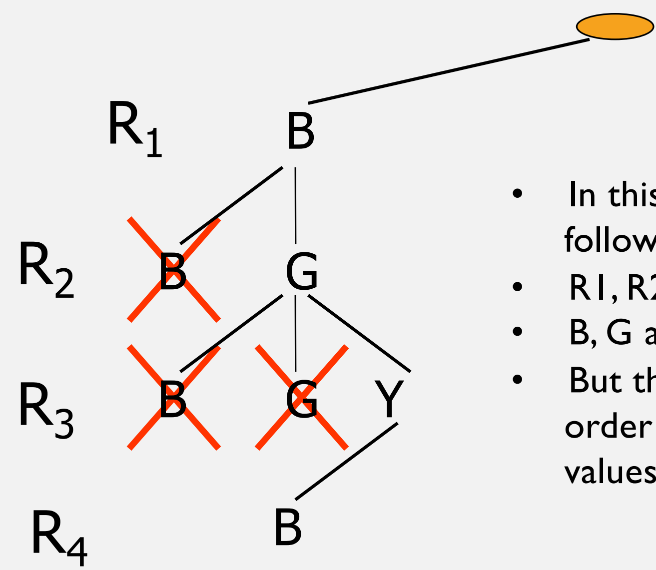
```
domz1(1).
domz2(1). domz2(2).
domz3(1). domz3(2). domz3(3).
domz4(1). domz4(2). domz4(3). domz4(4).


solution(Z1,Z2,Z3,Z4) :-
   domz1(Z1),
   domz2(Z2), Z1 =< Z2,
   domz3(Z3), Z2 =< Z3,
   domz4(Z4), X is Z2 * Z4, X > 1,
                       Y is Z4 + 2, Z3 = Y.
```



GENERAL PURPOSE HEURISTICS

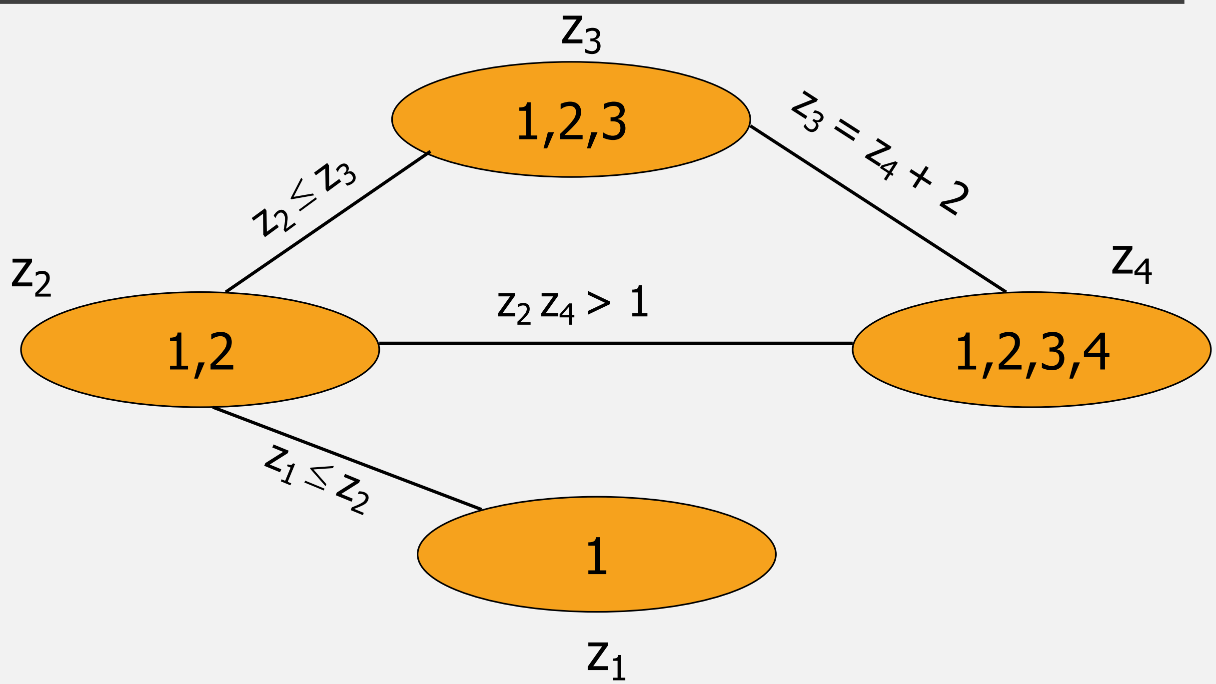
- ¿In what order we should visit the variables?
- ¿In what order we should assign the values of each variable?

VARIABLE AND VALUE ORDERINGS



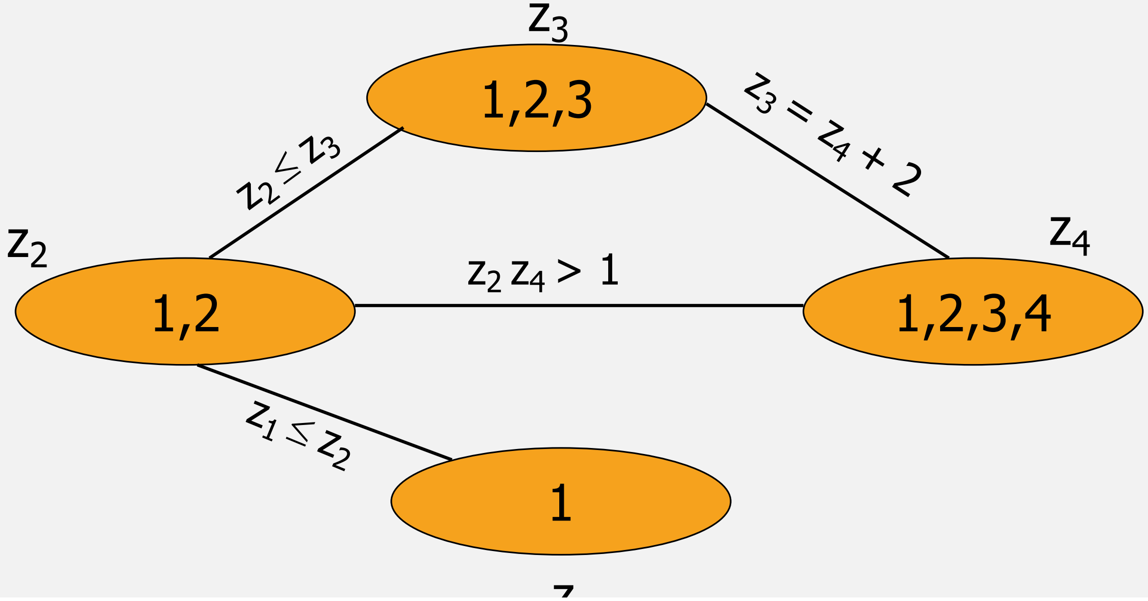
- In this example we used the following orderings:
- R1, R2, R3 and R4
- B, G and Y
- But this is the best way to order these variables and values?

HOW DO YOU SOLVED THIS PROBLEM?



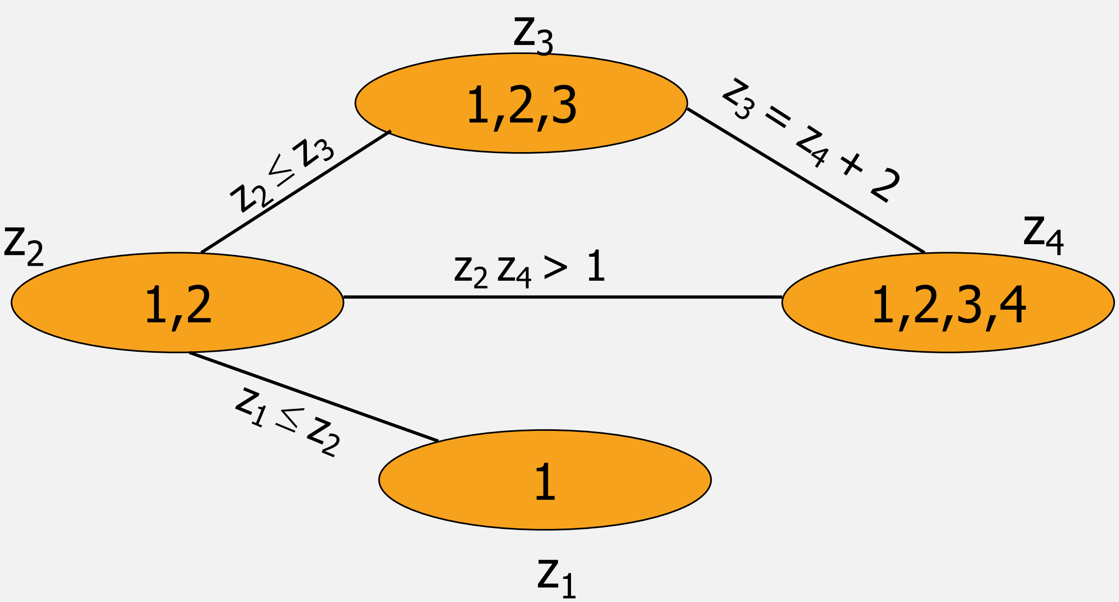
HEURISTIC 1: "MINIMUM REMAINING VALUES (MVR)" (IT HAS ALSO BEEN CALLED "MOST-CONSTRAINED-VARIABLE" OR "FAIL-FIRST" HEURISTIC)

- To pick first the variable with the smallest domain: {z1,z2,z3,z4}



HEURISTIC 2: DEGREE HEURISTIC

- To pick first the variable that in involved in the largest number of constraints: {z2,z4,z3,z1} o {z2,z3,z4,z1}



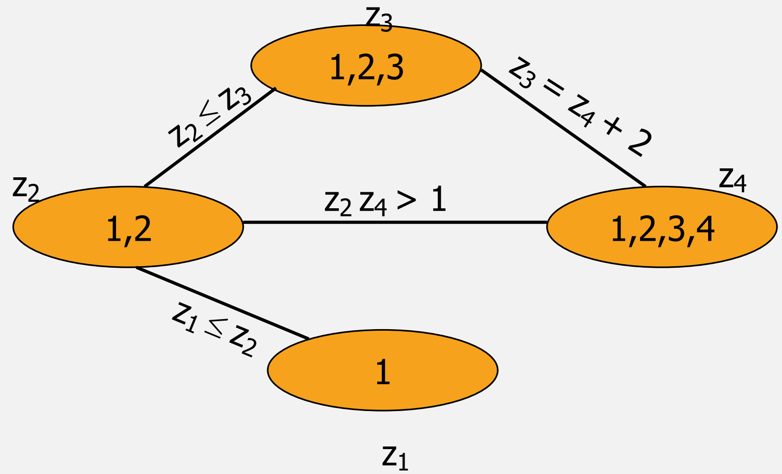
HEURISTIC 3

- Relaxation factor:

$$R_c = \frac{card(T_c)}{card(\times d_{x_c})}$$

T_c = pairs that satisfy the restriction
d_{x_c} = size of the domain of the involved variables

HEURISTIC 3



- $R(z2 \leq z3) = 5/6 = 83\%$
- $R(z2\ z4 > 1) = 7/8 = 87\%$
- $R(z1 \leq z2) = 2/2 = 100\%$
- $R(z3 = z4 + 2) = 1/12 = 8\%$
- Order: {z4,z3,z2,z1} o {z3,z4,z2,z1}

NOW ORDERING VALUES

- Now, we have three heuristics for selecting the order of variables.
- Buy what about the order of the values of each variable?
- **Heuristic “least-constraining-value”**: To choose first the value that rules out the fewest choices for the neighbouring variables in the constraint graph.

EXAMPLE WITH THE 8-QUEENS PROBLEM

- Place n queens in a chees board of $n \times n$ cells such that no queen attacks any other.
- Strategy: Each queen domain is a column.
- Variables: {R1,R2,R3,R4 ,R5,R6,R7 ,R8}
- Dominios: {1,2,3,4,5,6,7,8}
- Restricciones: non-attack(Ri,Rj)

PROBLEM WITH THE CLASSIC SEARCH

- It does not predict the future!

R							
			R				
	R						
		R					

With this configuration, you can not place a queen in the sixth column!

“FORWARD-CHECKING (FCH)”

- Whenever a variable X is assigned, the forward checking process establishes arc consistency for it: for each unassigned variable Y that is connected to X by constraint, delete from Y’s domain any value that is not consistent with the value chosen for X.
- If a domain is empty, we need to immediately backtrack!

FORWARD CHECKING

R	X	X	X	X	X	X	X
	X	X	R	X	X	X	X
	R	X	X	X	X	X	X
		X	X		X		
		R	X	X	X	X	X
			X	X	X		X
				X	X	X	
					X	X	X

FORWARD CHECKING WITH "MRV + LCV"

8	8	8	8	8	8	8	8
14							
14							
14							
14							
14							
14							
14							
14							

FORWARD CHECKING WITH "LEAST-CONSTRAINING VALUE"

	6	6	6	6	6	6	6
R	X	X	X	X	X	X	X
	X						
	11	X					
	10		X				
	11			X			
	10				X		
	11					X	
	10						X

MRV + LCV + FCH

		4	4	4	4	5	5
R	X	X	X	X	X	X	X
	X	8	X				
		X					
	R	X	X	X	X	X	X
		X		X			
		8	X		X		
		6		X		X	
		7			X		X

MRV + LCV + FCH

			2	4	3	4	3
R	X	X	X	X	X	X	X
	X		X				X
		X	6			X	
	R	X	X	X	X	X	X
		X	7	X			
			X		X		
		R	X	X	X	X	X
			X		X		X

OTHER OPTIONS TO BACKTRACKING

- Other strategies:
  - **Backjumping**
  - **Conflict-directed backjumping**
  - **Constraint learning**
  - **Etc …**

ARC CONSISTENCY

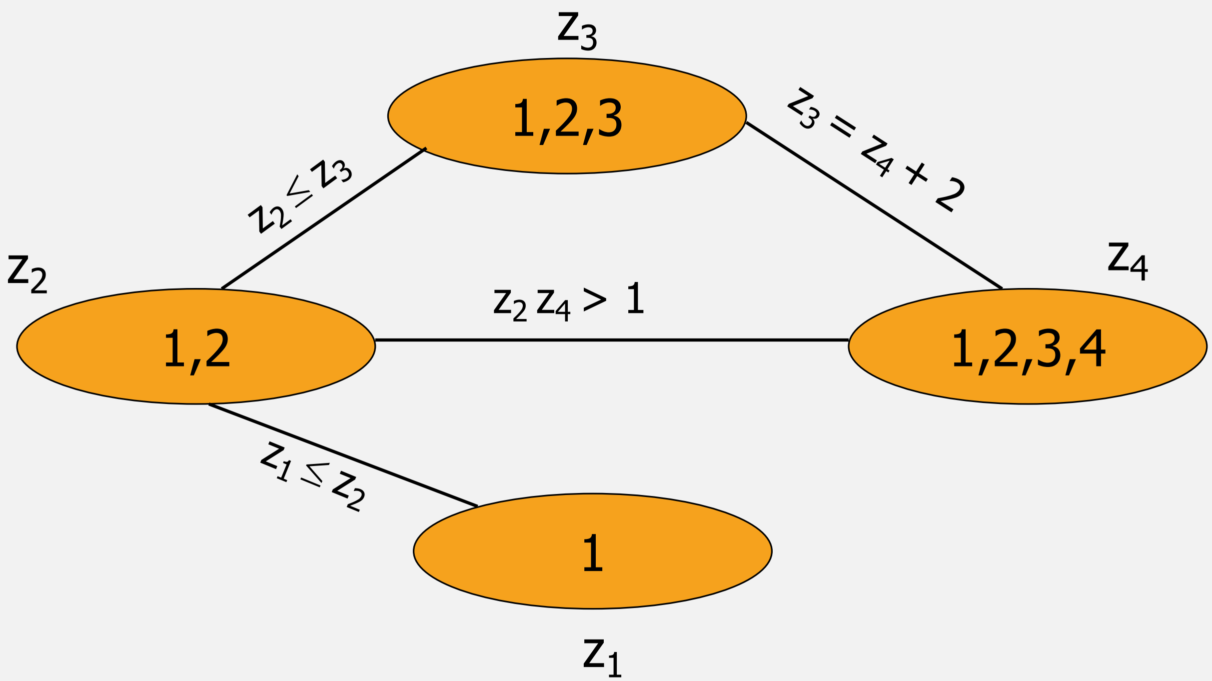
- A variable in a CSP is "**arc-consistent**" if every value in its domain satisfies the variable's binary constraints.
- More formally: Xi is **"arc-consistent"** with respecto to another variable Xj, if for every value in domain Di there is some value in domain Dj that satisfies the binary constraint in the arc (Xi, Xj).
- This idea is applied in AC-3 algorithm of Mackworth: **Constraint Propagation using Arc Consistency of Arcs**

ALGORITHM AC-3

function AC-3(csp) **return** the CSP, possibly with reduced domains
inputs: csp, a binary csp with variables $\{X_1, X_2, \ldots, X_n\}$
local variables: queue, a queue of arcs initially the arcs in csp
while queue is not empty **do**
$(X_i, X_j) \leftarrow$ REMOVE-FIRST(queue)
if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
for each X_k **in** NEIGHBORS[X_i] **do**
add (X_k, X_i) to queue

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **return** true iff we remove a value
removed $\leftarrow$ false
for each x **in** DOMAIN[X_i] **do**
if no value y in DOMAIN[X_j] does (x, y) satisfy constraints between X_i and X_j
then
delete x from DOMAIN[X_i];
removed $\leftarrow$ true
return removed

AC-3



AFTER RUNNING AC-3