

# Super Solutions

Emmanuel Hebrard

The logo for LAAS-CNRS features the text "LAAS-CNRS" in a dark blue, sans-serif font. The text is centered between two horizontal lines: a purple line above and a yellow line below.

LAAS-CNRS

Toulouse

# Outline

- 1 **Constraint Programming**
- 2 **Robustness**
- 3 **Super Solutions**
- 4 **Applications**

# Outline

**1 Constraint Programming**

2 Robustness

3 Super Solutions

4 Applications

# Constraint Satisfaction Problem

- Variables: finite discrete domain ( $\subseteq \mathbb{Z}$ )
- Constraints: any polynomial-time checkable relation
  - ▶ Logical or arithmetic operators  $\{\neq, >, \leq, \text{or}, \Rightarrow, \dots\}$
  - ▶ Linear or non-linear equations
  - ▶ Standard subproblems
    - ★ Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - ★ NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation!**
  - ▶ Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

## Constraint Satisfaction Problem

- Variables: finite discrete domain ( $\subseteq \mathbb{Z}$ )
- Constraints: any polynomial-time checkable relation
  - ▶ Logical or arithmetic operators  $\{\neq, >, \leq, \text{or}, \Rightarrow, \dots\}$
  - ▶ Linear or non-linear equations
  - ▶ Standard subproblems
    - ★ Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - ★ NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation!**
  - ▶ Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

$$\begin{array}{ccccc} x & \xrightarrow{<} & y & \xrightarrow{\neq} & z \\ \{1, 2, 3\} & & \{1, 2\} & & \{1, 2, 3\} \end{array}$$

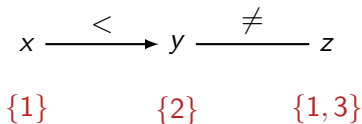
## Constraint Satisfaction Problem

- Variables: finite discrete domain ( $\subseteq \mathbb{Z}$ )
- Constraints: any polynomial-time checkable relation
  - ▶ Logical or arithmetic operators  $\{\neq, >, \leq, \text{or}, \Rightarrow, \dots\}$
  - ▶ Linear or non-linear equations
  - ▶ Standard subproblems
    - ★ Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - ★ NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation!**
  - ▶ Instead of finding a solution to these constraints, we look for inconsistent values, and remove them

$$\begin{array}{ccccc} x & \xrightarrow{<} & y & \xrightarrow{\neq} & z \\ \{1\} & & \{2\} & & \{1, 2, 3\} \end{array}$$

## Constraint Satisfaction Problem

- Variables: finite discrete domain ( $\subseteq \mathbb{Z}$ )
- Constraints: any polynomial-time checkable relation
  - ▶ Logical or arithmetic operators  $\{\neq, >, \leq, \text{or}, \Rightarrow, \dots\}$
  - ▶ Linear or non-linear equations
  - ▶ Standard subproblems
    - ★ Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - ★ NP-hard: Hitting set, Bin packing, Linear equality, ...
- Inference mechanism: **Propagation!**
  - ▶ Instead of finding a solution to these constraints, we look for inconsistent values, and remove them



# Outline

## 1 Constraint Programming

## 2 Robustness

- Context Free Robustness?
- Stability
- Super solutions

## 3 Super Solutions

## 4 Applications

## Solution Robustness

- Satisfaction, Optimisation: find a solution
- Uncertainty
  - ▶ Unexpected change in the data

### Change

A change can be seen as an additional constraint

### Solution Robustness

A solution  $\sigma$  is more robust than  $\sigma'$  iff the probability that a change invalidates  $\sigma$  is lower than the probability that it invalidates  $\sigma'$

## Context Free Robustness?

### Solution Robustness

A solution  $\sigma$  is more robust than  $\sigma'$  iff the probability that a change invalidates  $\sigma$  is lower than the probability that it invalidates  $\sigma'$

## Context Free Robustness?

### Solution Robustness

A solution  $\sigma$  is more robust than  $\sigma'$  iff the probability that a change invalidates  $\sigma$  is lower than the probability that it invalidates  $\sigma'$

- **Is it possible to characterise solution robustness without assumptions on the changes?**

# Context Free Robustness?

## Solution Robustness

A solution  $\sigma$  is more robust than  $\sigma'$  iff the probability that a change invalidates  $\sigma$  is lower than the probability that it invalidates  $\sigma'$

- **Is it possible to characterise solution robustness without assumptions on the changes?**
  - ▶ Not with the definition above
    - ★ There are exactly as many changes that invalidate  $\sigma$  and not  $\sigma'$  as the opposite
    - ★ Requires a probability distribution, or any kind of information on the possible changes

## Stability

- Consider the following Boolean satisfaction problem:

$$\sum_{i=1}^n x_i \geq k$$

- $111\dots 11$  or  $111100\dots 00$ ?
- the solution assigning every variable to  $1$  seems more “robust” than assigning only  $\{x_1, \dots, x_k\}$  to  $1$ :
  - ▶ If the change involves less than  $n - k$  variables, then simply re-assigning these variables in any consistent way yields a new solution
  - ▶ With the second solution, we may need to re-assign variables that were not involved in the change

## Stability

- Consider the following Boolean satisfaction problem:

$$\sum_{i=1}^n x_i \geq k$$

- 111...11 or 111100...00?
- the solution assigning every variable to 1 seems more “robust” than assigning only  $\{x_1, \dots, x_k\}$  to 1:
  - If the change involves less than  $n - k$  variables, then simply re-assigning these variables in any consistent way yields a new solution
  - With the second solution, we may need to re-assign variables that were not involved in the change

### Stability

A solution  $\sigma$  is more stable than  $\sigma'$  iff a change requires less repairs than on  $\sigma'$  to obtain a solution consistent with the change

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$\sigma$  1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$$\begin{array}{cccccccccccccccc} \sigma & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta & & & & & & & \underbrace{0 & 0 & 0 & 1}_{\text{break}} & & & & & & & & \end{array}$$

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$$\begin{array}{l} \sigma \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\ \beta \ 1 \ 1 \ 1 \ \underbrace{0 \ 0 \ 0 \ 1}_{\text{break}} \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{array}$$

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$$\begin{array}{l} \sigma \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\ \beta \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\ \quad \quad \quad \underbrace{\hspace{3cm}}_{\text{break}} \quad \underbrace{\hspace{3cm}}_{\text{repair}} \end{array}$$

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$$\begin{array}{cccccccccccccccc} \sigma & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta & 1 & 1 & \underbrace{0 & 0 & 0 & 0}_{\text{break}} & \underbrace{1 & 1 & 1 & 1}_{\text{repair}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

(worst case)

## Super Solutions

- Let  $\Phi$  be a problem,  $\sigma$  be a solution and  $c$  be a change
- We identify  $c$  with the set of variables that need to be changed in  $\sigma$  to satisfy  $c$ 
  - ▶ We call this set of variables a break and denote its size by  $a$
  - ▶ We call any solution of  $\Phi$  &  $c$  a repair, and denote the size of its difference with respect to  $\sigma$  by  $b$

$$\sum_{i=1}^n x_i \geq 6$$

$$\begin{array}{l} \sigma \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\ \beta \ 1 \ 1 \ \underbrace{0 \ 0 \ 0 \ 0}_{\text{break}} \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \end{array}$$

## Super Solutions

### Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution  $\sigma$  is a  $(a, b)$ -super solution iff for all breaks of size  $a$  or less, there exists a repair of size  $b$  or less.

## Super Solutions

### Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution  $\sigma$  is a  $(a, b)$ -super solution iff for all breaks of size  $a$  or less, there exists a repair of size  $b$  or less.

- $1^n$  is a  $(a, 0)$ -super solution of  $\sum_{i=1}^n x_i \geq k$  for all  $a \leq n - k$

## Super Solutions

### Super Solution (SAT supermodels [Ginsberg, Parkes and Roy 1998])

The solution  $\sigma$  is a  $(a, b)$ -super solution iff for all breaks of size  $a$  or less, there exists a repair of size  $b$  or less.

- $1^n$  is a  $(a, 0)$ -super solution of  $\sum_{i=1}^n x_i \geq k$  for all  $a \leq n - k$
- Four Ph.D. Thesis on the topic!
  - ▶ Amitabha Roy “Symmetry Breaking and Fault Tolerance in Boolean Satisfiability” (1998)
  - ▶ Emmanuel Hebrard “Robust Solutions for Constraint Satisfaction and Optimisation under Uncertainty” (2006)
  - ▶ Alan Holland “Risk Management for Combinatorial Auctions” (2005)
  - ▶ Victor Muñoz “Robustness in Resource Allocation Problems” (2010)

# Outline

1 Constraint Programming

2 Robustness

**3 Super Solutions**

- Complexity
- Getting Super solutions

4 Applications

## Super-CSP is intractable

### CSP

Given a CSP  $\phi$ , does  $\phi$  admit a solution?

### Super-CSP

Given a CSP  $\phi$  and two ints  $a, b$ , does  $\phi$  admit a  $(a, b)$ -super solution?

- NP-hard trivial reduction from CSP
- in NEXP: exponential number of witnesses, each NP-hard to check

## Super-CSP is intractable

### CSP

Given a CSP  $\Phi$ , does  $\Phi$  admit a solution?

### Super-CSP

Given a CSP  $\Phi$  and two ints  $a, b$ , does  $\Phi$  admit a  $(a, b)$ -super solution?

- NP-hard trivial reduction from CSP
- in NEXP: exponential number of witnesses, each NP-hard to check

### Super solution checking

Given a CSP  $\Phi$ , a sol  $\sigma$  and an int  $b$ , is  $\sigma$  a  $(1, b)$ -super solution of  $\Phi$ ?

## Super-CSP is intractable

### CSP

Given a CSP  $\phi$ , does  $\phi$  admit a solution?

### Super-CSP

Given a CSP  $\phi$  and two ints  $a, b$ , does  $\phi$  admit a  $(a, b)$ -super solution?

- NP-hard trivial reduction from CSP
- in NEXP: exponential number of witnesses, each NP-hard to check

### Super solution checking

Given a CSP  $\phi$ , a sol  $\sigma$  and an int  $b$ , is  $\sigma$  a  $(1, b)$ -super solution of  $\phi$ ?

- NP-complete, reduction from K-Clique

## (a,b)-Super-CSP is NP-hard

### (a,b)-Super-CSP

Given a CSP  $\Phi$ , does  $\Phi$  admit a  $(a, b)$ -super solution?

- Membership to NP (witnessed by itself AND the polynomial set of repairs)
- If  $(1, b)$ -Super-CSP is NP-hard then  $(1, b+1)$ -Super-CSP is NP-hard
  - ▶ Hence NP-complete for all  $a, b$

## Finding Super Solution of Tractable-CSP

- SAT tractable classes [Ginsberg, Parkes and Roy 2006]
  - ▶ (1,b)-Horn-SAT is NP-complete for all  $b$
  - ▶ (1,b)-2-SAT is polynomial for  $b \leq 1$  and NP-complete otherwise
  - ▶ (a,b)-Affine-SAT is polynomial for all  $a, b$
- CSP tractable classes [Hebrard, Hnich and Walsh 2006]
  - ▶ (a,b)-Class-0-CSP is NP-hard for all  $a, b$
  - ▶ (1,b)-Tree-CSP is polynomial for  $b = 0$ , NP-complete for  $b > 1$  and open for  $b = 1$
  - ▶ (1,b)-Majority-CSP is NP-complete for  $b > 1$ , and open otherwise

## Finding Super Solutions

- Simplest case:  $(1, 0)$ -super solutions
  - ▶ For each variable, there is an alternative value for this variable

## Finding Super Solutions

- Simplest case: (1, 0)-super solutions
  - ▶ For each variable, there is an alternative value for this variable

$$x, y, z \in \{1, 2, 3\}$$

$$x \neq y, y \leq z$$

## Finding Super Solutions

- Simplest case:  $(1, 0)$ -super solutions
  - ▶ For each variable, there is an alternative value for this variable

$$x, y, z \in \{1, 2, 3\}$$

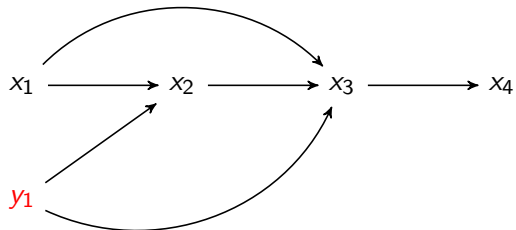
$$x \neq y, y \leq z$$

- $\langle x = 3, y = 1, z = 3 \rangle$  is a  $(1, 0)$ -super solution, since:
  - ▶  $\langle x = 2, y = 1, z = 3 \rangle$  is a solution
  - ▶  $\langle x = 3, y = 2, z = 3 \rangle$  is a solution
  - ▶  $\langle x = 3, y = 1, z = 2 \rangle$  is a solution

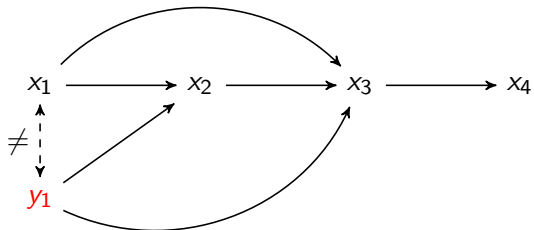
## Reformulation ( $\mathcal{P} + \mathcal{P}$ )



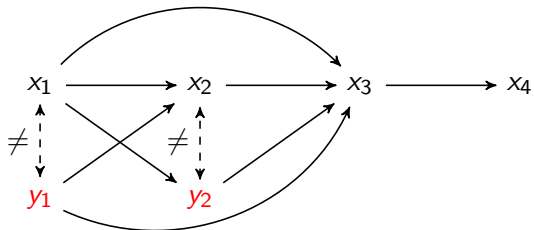
## Reformulation ( $\mathcal{P} + \mathcal{P}$ )



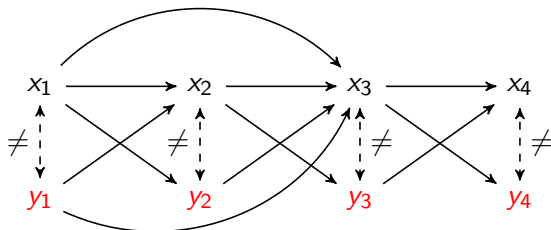
## Reformulation ( $\mathcal{P} + \mathcal{P}$ )



## Reformulation ( $\mathcal{P} + \mathcal{P}$ )

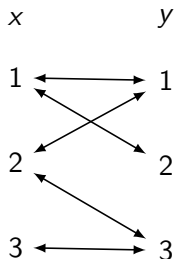


## Reformulation ( $\mathcal{P} + \mathcal{P}$ )



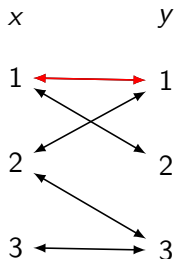
- Given a solution  $\sigma$ , its restriction to  $x_1, x_2, x_3, x_4$  is a  $(1, 0)$ -super solution
  - its restriction to  $y_1, x_2, x_3, x_4$  is a repair for the break  $\{x_1\}$
  - its restriction to  $x_1, y_2, x_3, x_4$  is a repair for the break  $\{x_2\}$
  - its restriction to  $x_1, x_2, y_3, x_4$  is a repair for the break  $\{x_3\}$
  - its restriction to  $x_1, x_2, x_3, y_4$  is a repair for the break  $\{x_4\}$

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



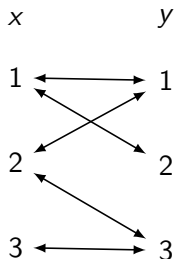
- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



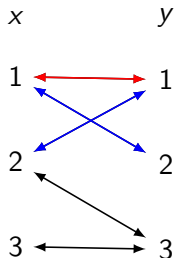
- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



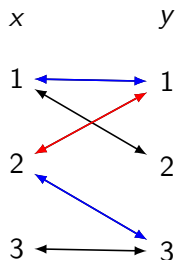
- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - ▶ And an alternative support

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



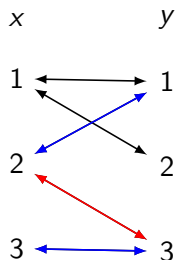
- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - ▶ And an alternative support

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



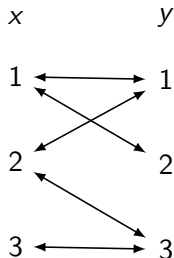
- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - ▶ And an alternative support

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - ▶ And an alternative support

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ ) / Super Arc-Consistency



- Arc-Consistency: Each value has a support
  - ▶ Local inconsistency  $\Rightarrow$  global inconsistency
- “Super” Arc-Consistency: Each value has a support
  - ▶ And an alternative support
- Reformulation with stronger propagation
- Reformulation of the algorithm for Arc-Consistency

# Outline

1 Constraint Programming

2 Robustness

3 Super Solutions

**4 Applications**

## Combinatorial Auction [Holland & O'Sullivan]

## Combinatorial Auction [Holland & O'Sullivan]

- 3 pictures to sell A:



B:



C:



## Combinatorial Auction [Holland & O'Sullivan]

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	200000	250000	250000	250000	250000
Mouhamad	100000	100000	100000	300000	200000	300000	400000
Ziad	50000	50000	100000	200000	150000	150000	400000

# Combinatorial Auction [Holland & O'Sullivan]

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	200000	250000	250000	250000	250000
Mouhamad	100000	100000	100000	300000	200000	300000	400000
Ziad	50000	50000	100000	200000	150000	150000	400000

- Winner Selection Problem

- ▶ A  $\{0, 1\}$  variable for each bid
- ▶  $\text{Bernard}_{A,B}$  takes the value 1 iff Bernard's bid on items  $A$  and  $B$  wins

- maximize: the sum of the bids
- subject to: no item is attributed more than once

# Combinatorial Auction

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	200000	250000	250000	250000	250000
Mouhamad	100000	100000	100000	300000	200000	300000	400000
Ziad	50000	50000	100000	200000	150000	150000	400000

# Combinatorial Auction

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	<u>200000</u>	0	200000	250000	250000	250000	250000
Mouhamad	100000	100000	100000	300000	200000	<u>300000</u>	400000
Ziad	50000	50000	100000	200000	150000	150000	400000

- (1) Bernard gets *A* and Mouhamad gets *B&C* (500000€)

# Combinatorial Auction

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	<u>200000</u>	0	200000	250000	250000	250000	250000
Mouhamad							
Ziad	50000	50000	100000	200000	150000	150000	400000

- (1) Bernard gets *A* and Mouhamad gets *B&C* (500000€)
  - Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or *A&B* to Ziad and *C* to Bernard (400000€)

# Combinatorial Auction

- 3 pictures to sell A:



- B:



- C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	200000	250000	250000	250000	250000
Mouhamad							
Ziad	50000	50000	100000	200000	150000	150000	400000

- (1) Bernard gets *A* and Mouhamad gets *B&C* (500000€)
  - Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or *A&B* to Ziad and *C* to Bernard (400000€)
    - ★ Either lose money or take back picture *A* from its winner (Bernard)

# Combinatorial Auction

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	<u>200000</u>	250000	250000	250000	250000
Mouhamad	100000	100000	100000	<u>300000</u>	200000	300000	400000
Ziad	50000	50000	100000	200000	150000	150000	400000

- (1) Bernard gets *A* and Mouhamad gets *B&C* (500000€)
  - ▶ Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or *A&B* to Ziad and *C* to Bernard (400000€)
    - ★ Either lose money or take back picture *A* from its winner (Bernard)
- (2) Bernard gets *C* and Mouhamad gets *A&B*

# Combinatorial Auction

- 3 pictures to sell A:



B:



C:



	A	B	C	AB	AC	BC	ABC
Bernard	200000	0	<u>200000</u>	250000	250000	250000	250000
Mouhamad	100000	100000	100000	<u>300000</u>	200000	300000	400000
Ziad	50000	50000	<u>100000</u>	<u>200000</u>	150000	150000	400000

- (1) Bernard gets *A* and Mouhamad gets *B&C* (500000€)
  - Mouhamad withdraw his bid: the best solutions are to give everything to Ziad, or *A&B* to Ziad and *C* to Bernard (400000€)
    - ★ Either lose money or take back picture *A* from its winner (Bernard)
- (2) Bernard gets *C* and Mouhamad gets *A&B*
  - Mouhamad withdraw his bid: Ziad replace him (400000€)
  - Bernard withdraw his bid: Ziad replace him (400000€)

## Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - ▶ Robustness, stochastic reasoning (**probabilistic super solutions**)

## Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - ▶ Robustness, stochastic reasoning (**probabilistic super solutions**)
- Tradeoff between stability and (expected) utility

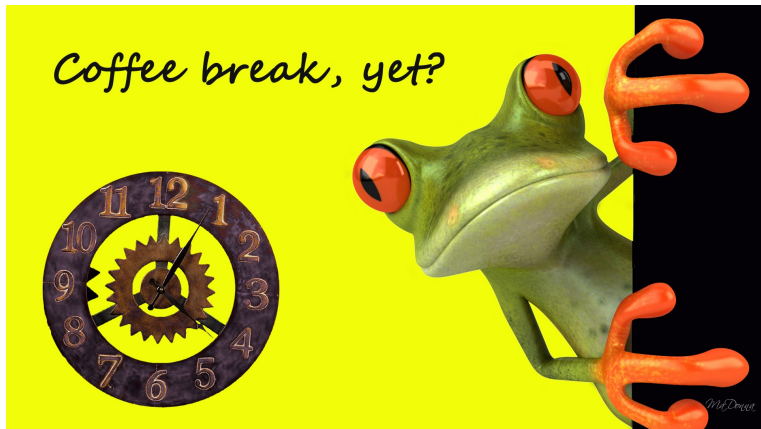
## Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - ▶ Robustness, stochastic reasoning (**probabilistic super solutions**)
- Tradeoff between stability and (expected) utility
- In Combinatorial auctions, impact of the stability on the truthfulness of the agents
  - ▶ A bid is truthful if it is a correct assessment of the utility of the item to the agent
  - ▶ Auction systems should incentivize truthful bidding
    - ★ In Vickrey auctions, truthfull bidding is a dominating strategy

## Other Results [Holland], [Muñoz]

- Take into account the probability of a break
  - ▶ Robustness, stochastic reasoning (**probabilistic super solutions**)
- Tradeoff between stability and (expected) utility
- In Combinatorial auctions, impact of the stability on the truthfulness of the agents
  - ▶ A bid is truthful if it is a correct assessment of the utility of the item to the agent
  - ▶ Auction systems should incentivize truthful bidding
    - ★ In Vickrey auctions, truthfull bidding is a dominating strategy
    - ★ Not anymore with stable Vickrey auctions! (though it is possible to make it so)

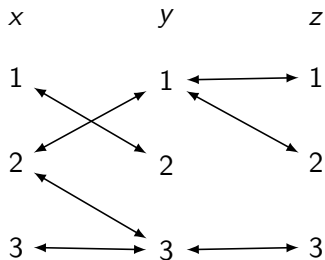
# Questions?



# Arc-Consistency

## Support

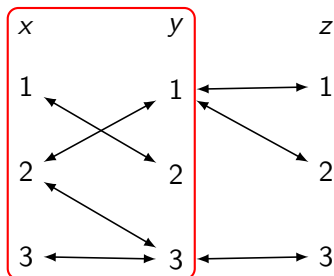
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

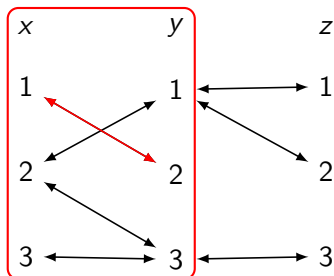
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

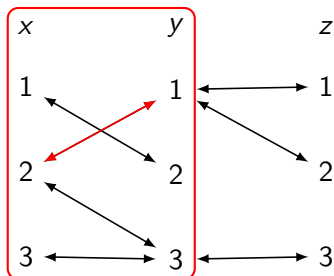
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

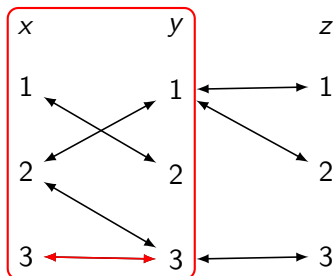
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

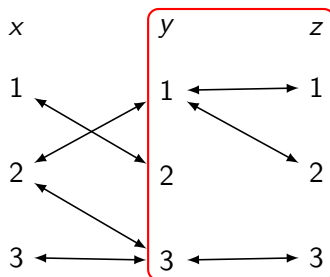
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

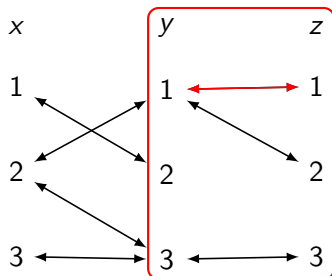
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

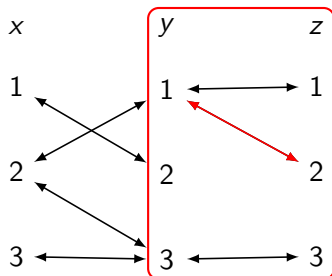
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

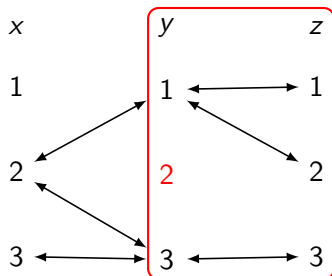
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

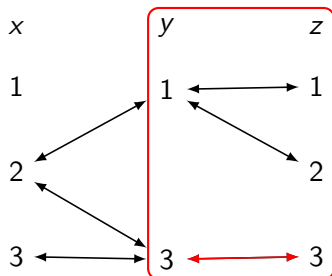
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

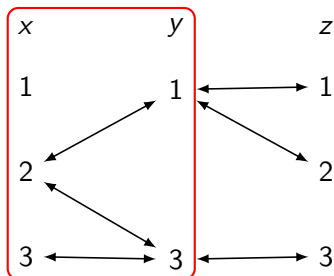
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



# Arc-Consistency

## Support

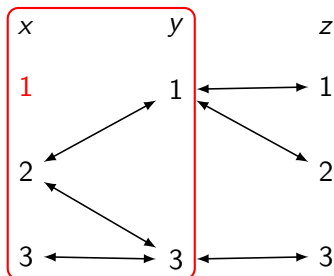
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



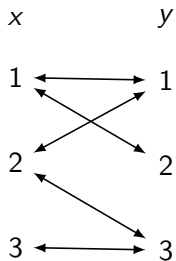
# Arc-Consistency

## Support

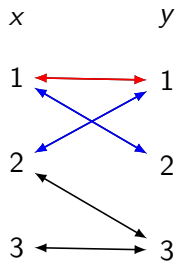
- A support  $\sigma$  of a value  $v$  for a constraint  $c$  is a solution of this constraint such that every value is itself arc-consistent
  - ▶ Propagation until reaching a fix point



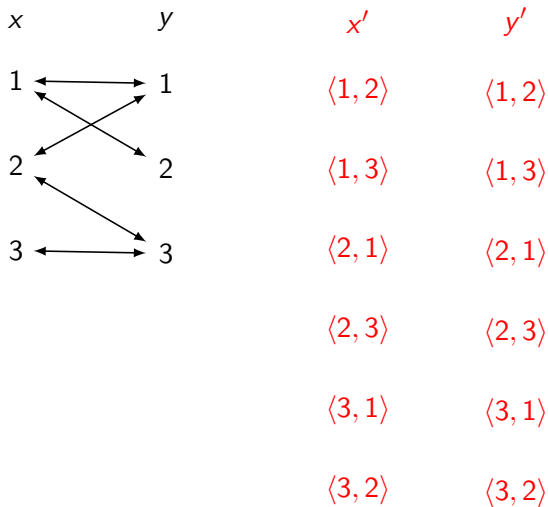
## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



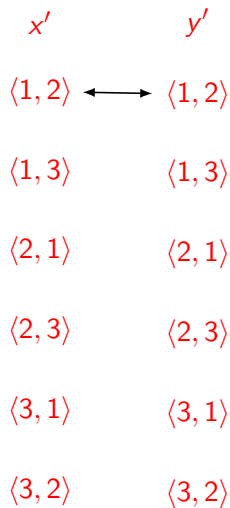
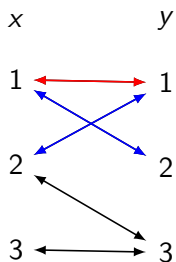
## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



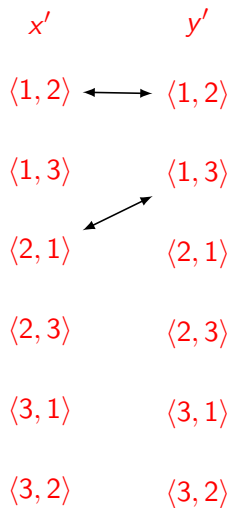
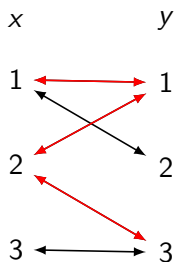
## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



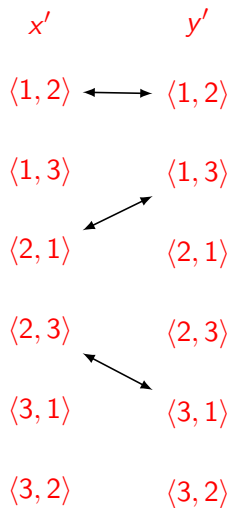
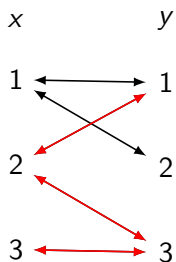
## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



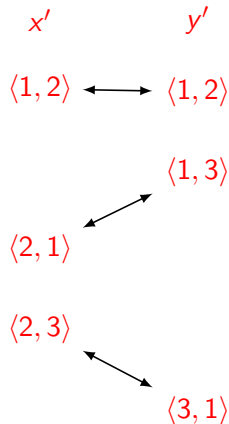
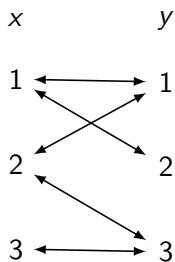
## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )

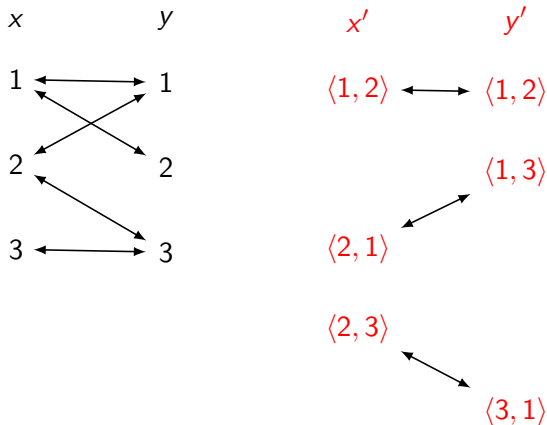


## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )



- This model allows stronger propagation!

## Reformulation ( $\mathcal{P} \times \mathcal{P}$ )

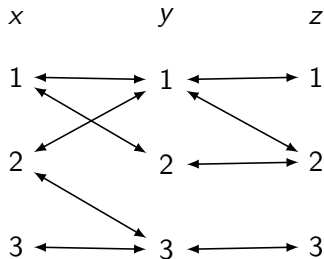


- This model allows stronger propagation!
- However, the domain size is quadratic

## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

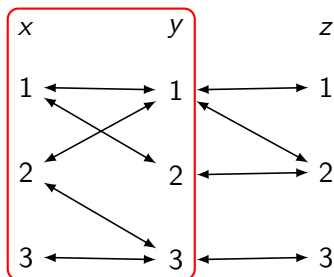
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

**“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain**

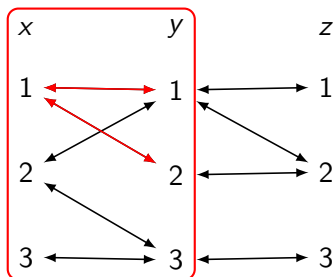
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

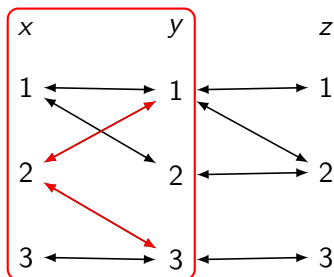
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

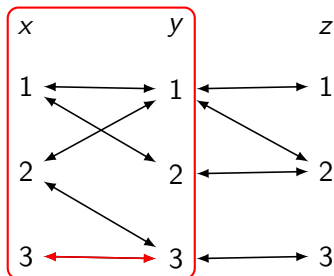
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

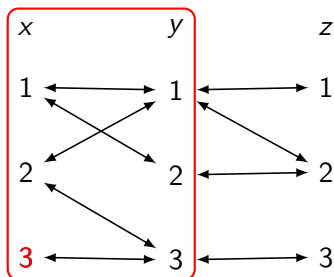
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

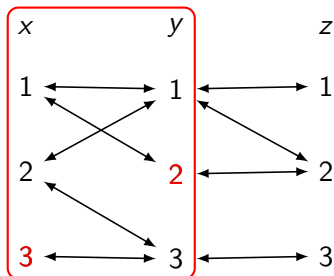
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

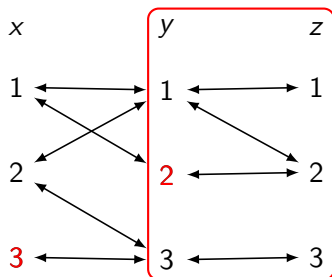
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

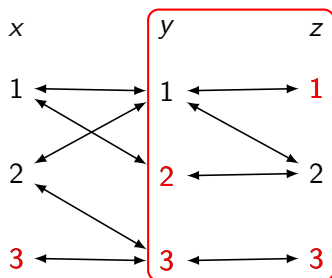
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

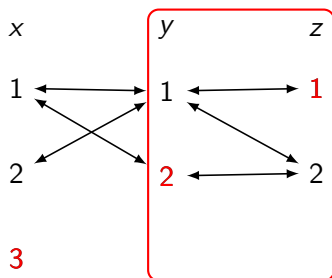
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

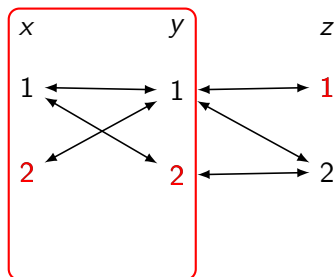
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

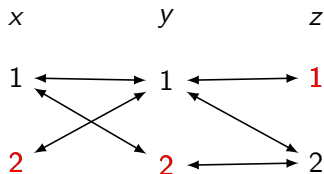
- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



## Super-Arc-Consistency

“super”-values  $\subseteq$  “repair”-values  $\subseteq$  domain

- Each “repair”-value must have a support in the set of “super”-values
- Each “super”-value must have a support in the set of “super”-values, and another in the set of “repair”-values



- The only  $(1, 0)$ -super solution is  $\langle 1, 1, 2 \rangle$

# Constraint Satisfaction Problem

- Variables: finite discrete domain ( $\subseteq \mathbb{Z}$ )
- Constraints: any polynomial-time checkable relation
  - ▶ Any fixed arity relation
    - ★ Logical or arithmetic operators  $\{\neq, >, \leq, \text{or}, \Rightarrow, \dots\}$
  - ▶ Linear or non-linear equations
  - ▶ Standard subproblems
    - ★ Polynomial: Matching, Sortedness, Cumulative Resource, ...
    - ★ NP-hard: Hitting set, Bin packing, Linear equality, ...

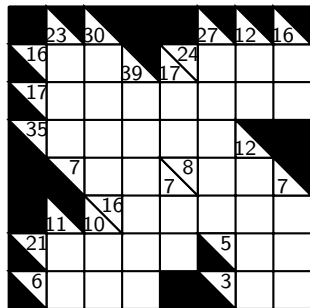
# CSP and Propagation

- Basic idea:
  - ▶ any NP-hard problem can be formulated as a conjunction of subproblems (constraints)
  - ▶ each constraint is easy (either polynomial or well understood)
- How to use that to solve the composite problem?
  - ▶ Propagation!

## Example: Kakuro

- $\sum_{i=1}^7 x_i = 39$
- $\text{MATCHING}(\{x_1, \dots, x_7\}, \{1, \dots, 9\})$

$x_1$  : { 8 9 }  
 $x_2$  : { 1 2 6 7 8 9 }  
 $x_3$  : { 8 9 }  
 $x_4$  : { 1 5 6 8 9 }  
 $x_5$  : { 1 2 6 7 8 9 }  
 $x_6$  : { 4 5 8 9 }



## Propagation

## Example: Kakuro

- $\sum_{i=1}^7 x_i = 39$
- $\text{MATCHING}(\{x_1, \dots, x_7\}, \{1, \dots, 9\})$

$x_1$  : { 8 9 }  
 $x_2$  : { 1 2 6 7 8 9 }  
 $x_3$  : { 8 9 }  
 $x_4$  : { 1 5 6 8 9 }  
 $x_5$  : { 1 2 6 7 8 9 }  
 $x_6$  : { 4 5 8 9 }



### Propagation

- $\text{MATCHING}(\{x_1, x_3\}, \{8, 9\})$









## Example: Kakuro

- $\sum_{i=1}^7 x_i = 39$
- $\text{MATCHING}(\{x_1, \dots, x_7\}, \{1, \dots, 9\})$

$$\begin{array}{l} x_1 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 8 \ 9 \} \\ x_2 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 6 \ 7 \} \\ x_3 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 8 \ 9 \} \\ x_4 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 5 \} \\ x_5 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 6 \ 7 \} \\ x_6 : \{ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 4 \ 5 \} \end{array}$$



### Propagation

- $\text{MATCHING}(\{x_2, x_5\}, \{6, 7\})$

