January 2022

Constraint Programming

Dr Ing. Houndji V. Ratheil
https://ratheil.info

Outline

General Introduction

Global Constraints

How CP works concretely ?

Take Away Message

Lecturer-Researcher at UAC

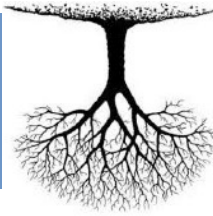
Houndji Vinasetan Ratheil,

Teaching experience (since my PhD thesis)

- Artificial Intelligence since 2014
- Optimisation since 2015
- Object Oriented Modelling/Programming
- Reliability of Information System
- etc.

Coordinator of Masters at IFRI, UAC

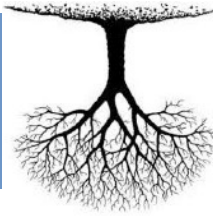
General Introduction



Constraint Programming (CP) is a paradigm derived from artificial intelligence, operational research, and algorithmic that can be used to solve combinatorial optimization problems.

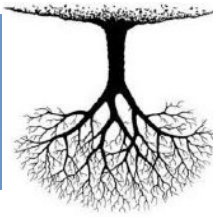
CP solves problems by interleaving search (assigning a value to an unassigned variable) and propagation (removing inconsistent values).

General Introduction



Constraint Programming allows a declarative description of the problem to solve.

General Introduction



Constraint Programming allows a declarative description of the problem to solve.

Other declarative approaches : **Logic Programming**, **Operations Research**

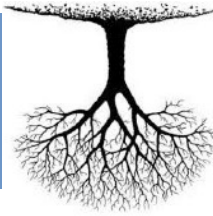
Find a formal representation of the knowledge with logical rules

A, B => G
D => A
C => H
I => F
H, F, J => B
H, K => J
C, F => K

Find a mathematical formulation of the problem

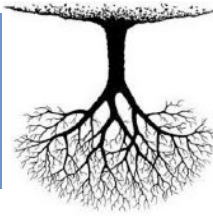
$$s_0^i = 0, \ \forall i$$
$$x_t^i + s_{t-1}^i = d_t^i + s_t^i, \forall i, t$$
$$x_t^i \leq y_t^i, \ \forall i, t$$
$$\sum_i y_t^i = 1, \forall t$$
$$\chi_t^{i,j} \geq y_{t-1}^i + y_t^j - 1, \forall i, j, t$$
$$x_t^i, y_t^i, \chi_t^{i,j} \in \{0,1\}, s_t^i \in \mathrm{N}, \forall i, j \in [1, \ldots, m]$$

General Introduction



In CP we would like to describe the problem in natural “english” language.

General Introduction



In CP we would like to describe the problem in natural “english” language.

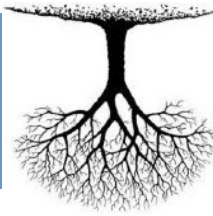
In practice it looks like

```
val nQueens = 8 // Number of queens
val Queens = 0 until nQueens

// Variables
val queens = Array.fill(nQueens)(CPIntVar.sparse( minValue = 0, nQueens - 1))

// Constraints
add(allDifferent(queens))
add(allDifferent(Queens.map(i => queens(i) + i)))
add(allDifferent(Queens.map(i => queens(i) - i)))
```

CP Applications

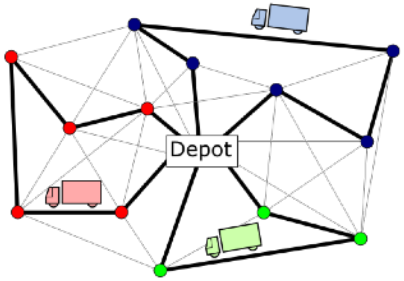


Production Planning

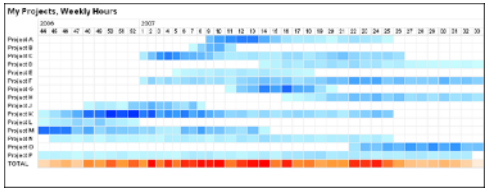
Agricultural land allocation

Rail Applications

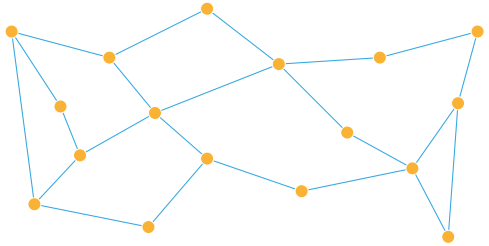
Vehicule Routing



Scheduling

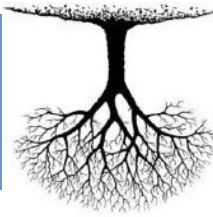


Trafic Engineering



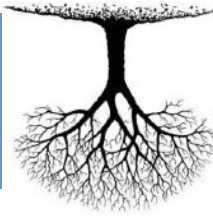
and much more…

A library of problems



A live demo with OscaR

A library of problems



CSPLib: A problem library for constraints

http://csplib.org/

058: Discrete Lot Sizing Problem

Specification Data files Results References Models Cite Json Edit Page

Proposed by Vinasetan Ratheil Houndji, Pierre Schaus, Laurence Wolsey, Yves Deville

Discrete Lot Sizing and Scheduling Problem (DLSP) is a production planning problem which consists of determining a minimal cost production schedule (production costs, setup costs, changeover costs, stocking costs, etc.), such that machine capacity restrictions are not violated, and demand for all products is satisfied. The planning horizon is discrete and finite.

The variant described here is the one used for experiments in The StockingCost Constraint.

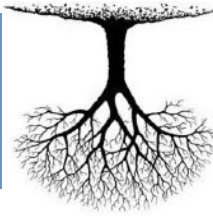
It is a multi-item, single machine problem with capacity of production limited to one per period. There are storage costs and sequence-dependent changeover costs, respecting the triangle inequality. Each order consisting of one unit of a particular item has a due date and must be produced at latest by its due date. The stocking (inventory) cost of an order is proportional to the number of periods between the due date and the production period. The changeover cost q^{ij} is induced when passing from the production of item i to another one j with $q^{i,i} = 0, \forall i$. Here, backlogging is not allowed. The objective is to assign a production period for each order respecting its due date and the machine capacity constraint so as to minimize the sum of stocking costs and changeover costs.

Example : Consider the problem with the following input data: number of items type $nbItems = 2$; number of periods $nbPeriods = 5$; stocking cost $h = 2$; demand times for items of type 1 $d^1_{i\in 1,\ldots,5} = (0, 1, 0, 0, 1)$ and for items of type 2 $d^2_{i\in 1,\ldots,5} = (1, 0, 0, 0, 1)$; $q^{1,2} = 5$, $q^{2,1} = 3$. A feasible solution of this problem is $productionPlan = (2, 1, 2, 0, 1)$ which means that item 2 will be produced in period 1; item 1 in period 2; item 2 in period 3 and item 1 in period 5. Note that there is no production in period 4, it is an idle period. The cost associated to this solution is $q^{2,1} + q^{1,2} + q^{2,1} + 2 * h = 15$ but it is not the optimal cost. The optimal solution is $productionPlan = (2, 1, 0, 1, 2)$ with the cost $q^{2,1} + q^{1,2} + h = 10$.

A Simulated Annealing metaheuristic approach along with a dataset of large-size instances has been developed by Sara Ceschia, Luca Di Gaspero and Andrea Schaerf. An online web solution checker and solution repository is available at https://opthub.uniud.it/problem/lsp.

This library presents a list of 92 problems
in 14 categories proposed by 70 authors.

General Introduction



$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

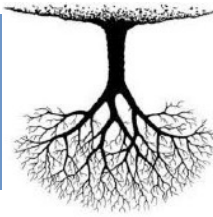
Solution of ***CSP***:

- An assignment of all variables
- All constraints are satisfiable

Model
+
Resolution

COP (Constraint Optimisation Problem):
CSP + objectif function(s)

General Introduction



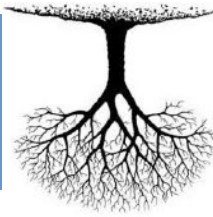
$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

Example: $X_1 \in \{1, 2\}$ $X_2 \in \{1, 2\}$ $X_1 \neq X_2$

A solution ?

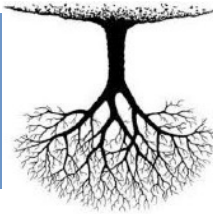
Modeling : an example



A farmer sends his son to the market with XOF100.000. With this XOF100.000, the son must buy a hundred animals. He has to come back to the farm with the 100 animals and he has to spend the full XOF100.000. A chick costs XOF500, a pig XOF5.000, and an ox XOF20.000. The son must buy at least one animal of each species.
How many chicks, pigs, and oxen will the son have?

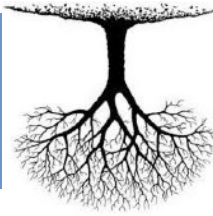
To do : propose a CP model for this problem.

General Introduction



Model
+
Resolution

General Introduction

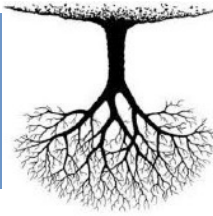


$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

Model
+
Resolution

General Introduction



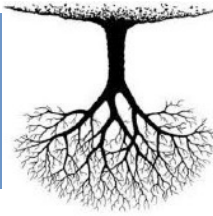
$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

Example $X_1 \in \{1, 2\}$ $X_2 \in \{1, 2\}$ $X_1 \neq X_2$

Model + Resolution

General Introduction



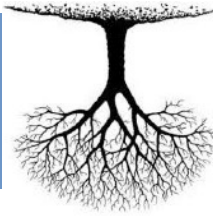
$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

Model + Resolution

Filtering + Search

General Introduction



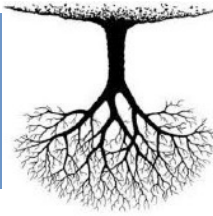
$$CSP = (X, D, C)$$

Variables X_i **Domains** $D(X_i)$ **Constraints**

Model + Resolution

Filtering + Search

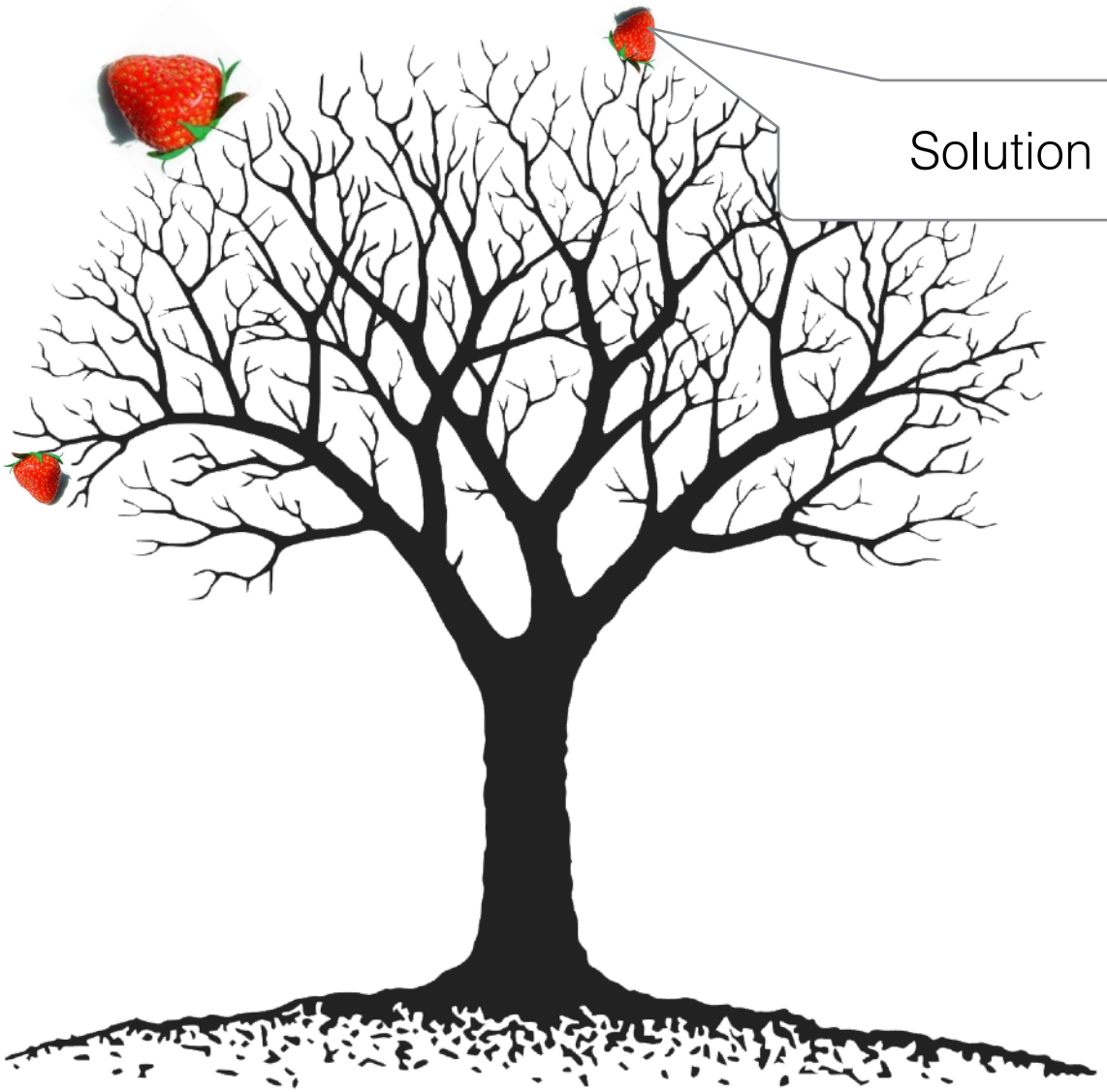
General Introduction



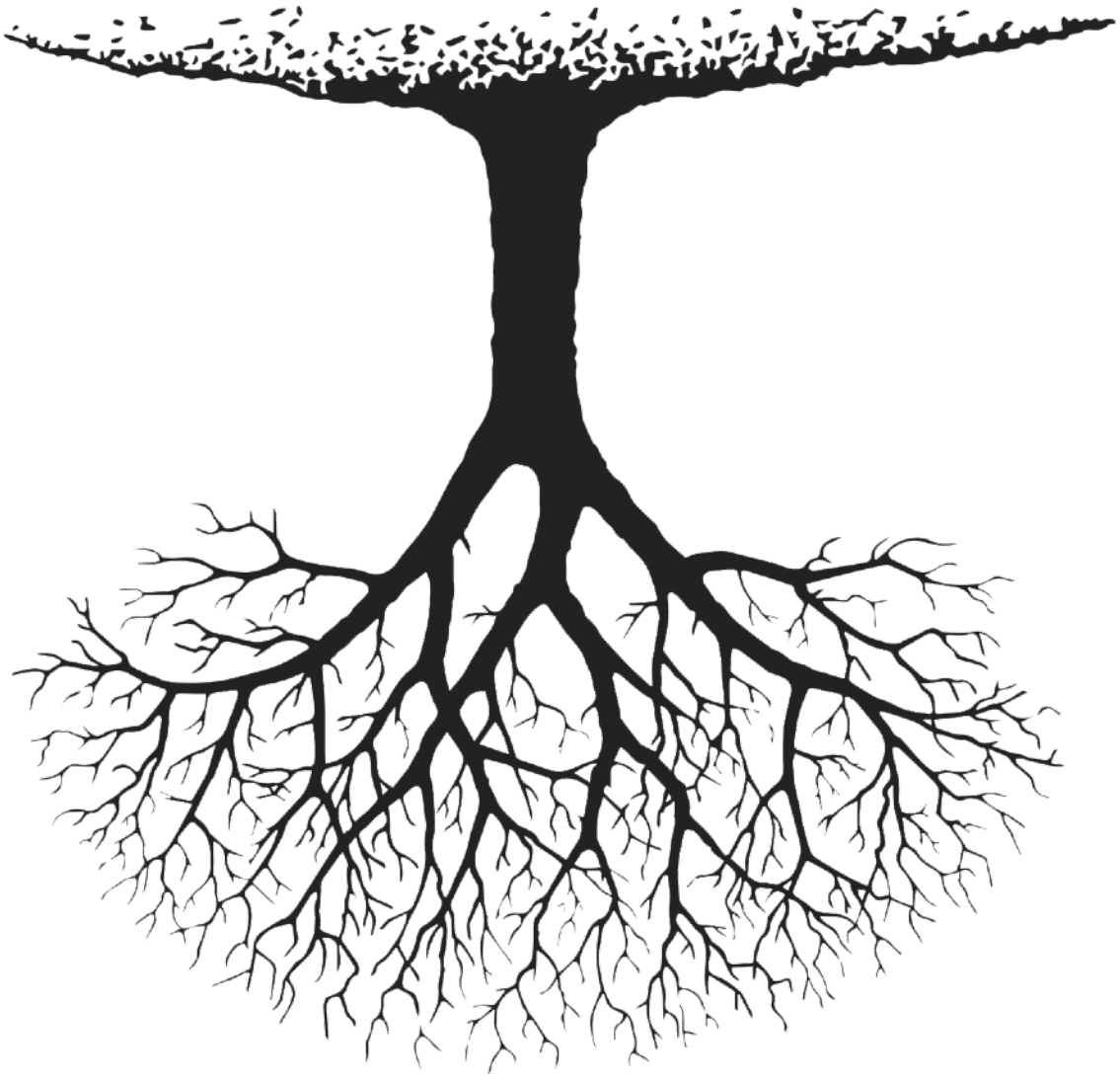
Resolution in CP : an intuition

Fruit seeking

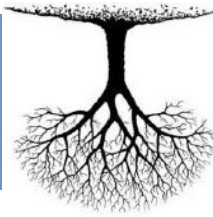
General Introduction



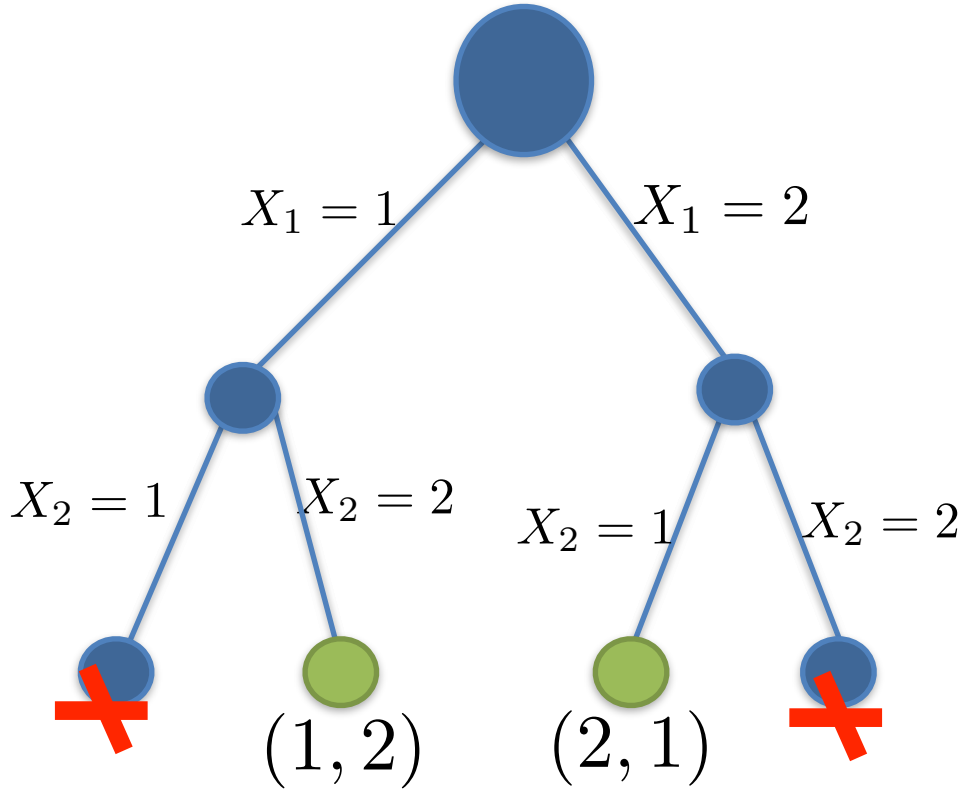
General Introduction



General Introduction

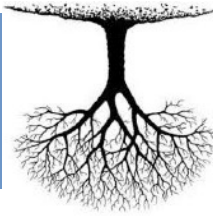


Example: $X_1 \in \{1,2\}\ X_2 \in \{1,2\}\quad X_1 \neq X_2$

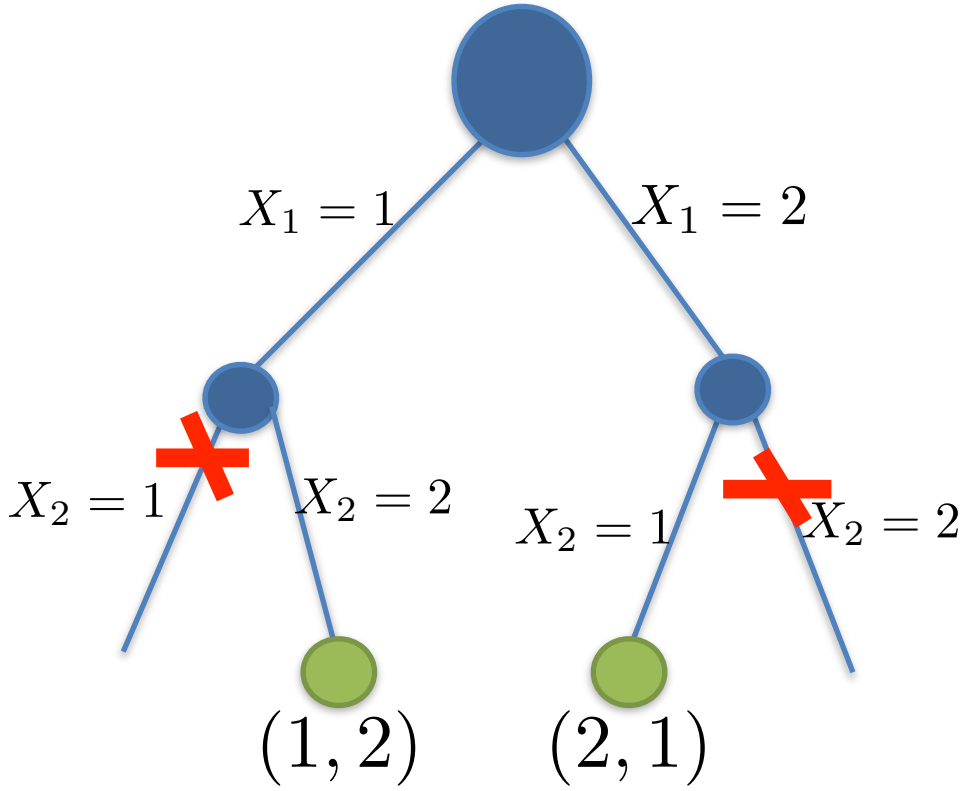


Search

General Introduction



Example : $X_1 \in \{1,2\}\ X_2 \in \{1,2\}\quad X_1 \neq X_2$

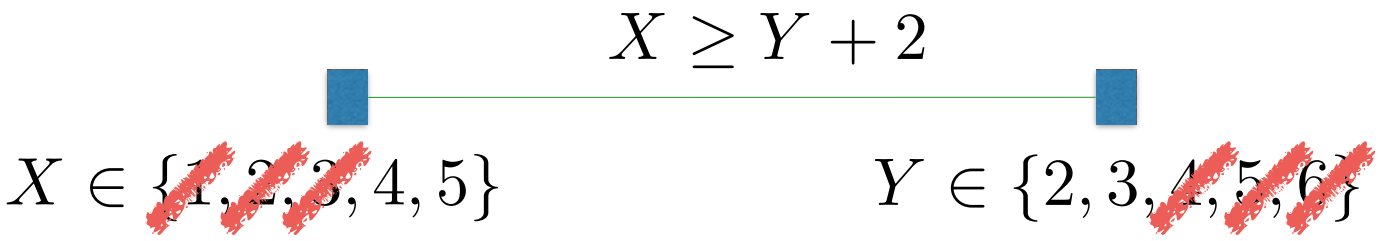


Filtering

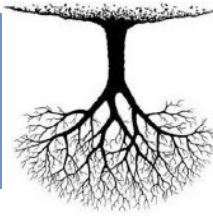
General Introduction

Filtering removes inconsistent values

Example 2:



General Introduction



$$\text{CP} = \text{Search} + \boxed{\text{Filtering}}$$

Too small

Too big

Outline

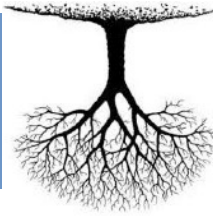
General Introduction

Global Constraints

How CP works concretely ?

Take Away Message

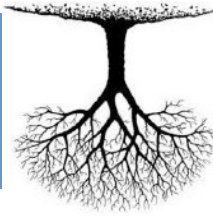
Global Constraints



Global constraint :

- A constraint reasoning on many variables at the same time
- Specialized, powerful filtering

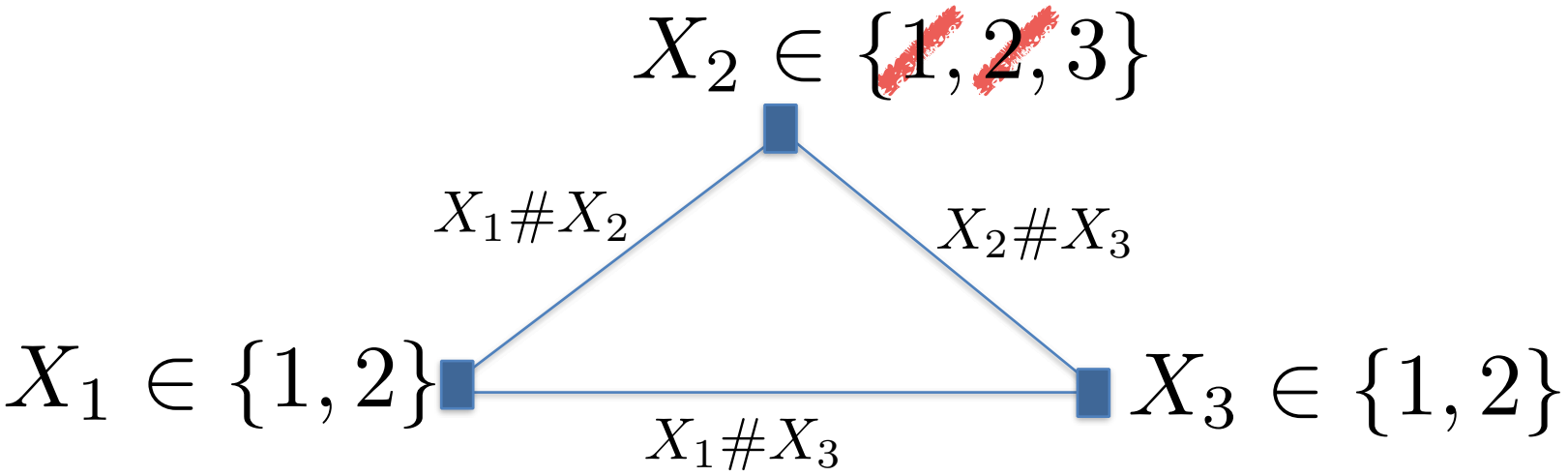
Global Constraints



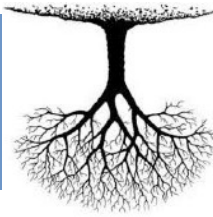
Global constraint :

- A constraint reasoning on many variables at the same time
- Specialized, powerful filtering

Example : $allDifferent(X_1, X_2, X_3)$

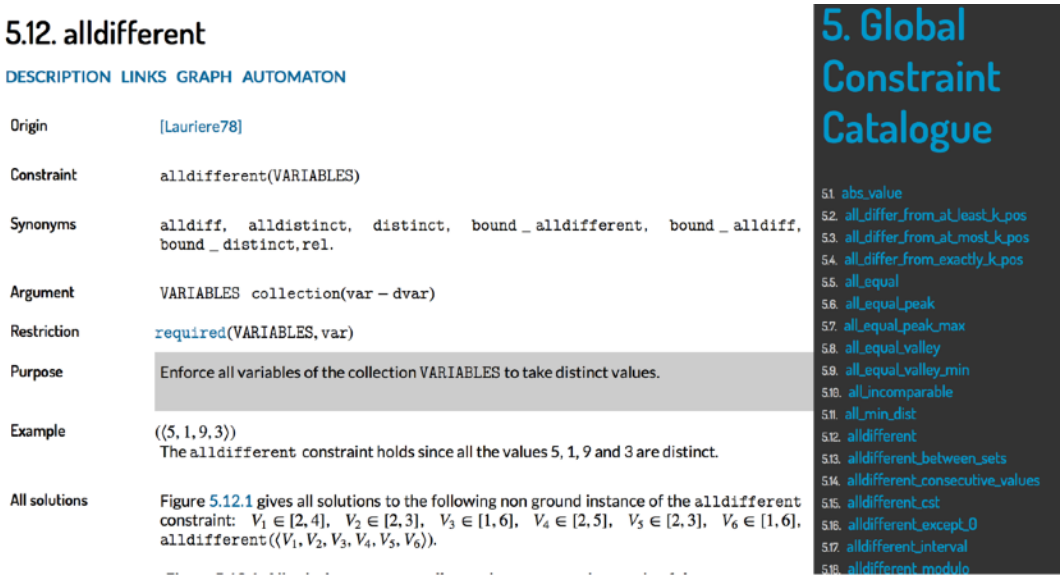


Global Constraints: Catalog



A catalog of more than 400 global constraints.

http://sofdem.github.io/gccat/



5.12. alldifferent

DESCRIPTION LINKS GRAPH AUTOMATON

Origin	[Lauriere78]
Constraint	alldifferent(VARIABLES)
Synonyms	alldiff, alldistinct, distinct, bound_alldifferent, bound_alldiff, bound_distinct,rel.
Argument	VARIABLES collection(var – dvar)
Restriction	required(VARIABLES, var)
Purpose	Enforce all variables of the collection VARIABLES to take distinct values.
Example	((5, 1, 9, 3)) The alldifferent constraint holds since all the values 5, 1, 9 and 3 are distinct.
All solutions	Figure 5.12.1 gives all solutions to the following non ground instance of the alldifferent constraint: $V_1 \in [2,4]$, $V_2 \in [2,3]$, $V_3 \in [1,6]$, $V_4 \in [2,5]$, $V_5 \in [2,3]$, $V_6 \in [1,6]$, alldifferent($\langle V_1, V_2, V_3, V_4, V_5, V_6 \rangle$).

This catalog presents a list of 423 global constraints
issued from the literature in constraint programming and from popular constraint systems.

Outline

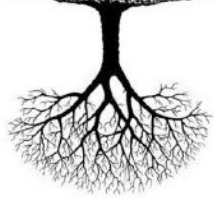
General Introduction

Global Constraints

How CP works concretely ?

Take Away Message

How CP works concretely?



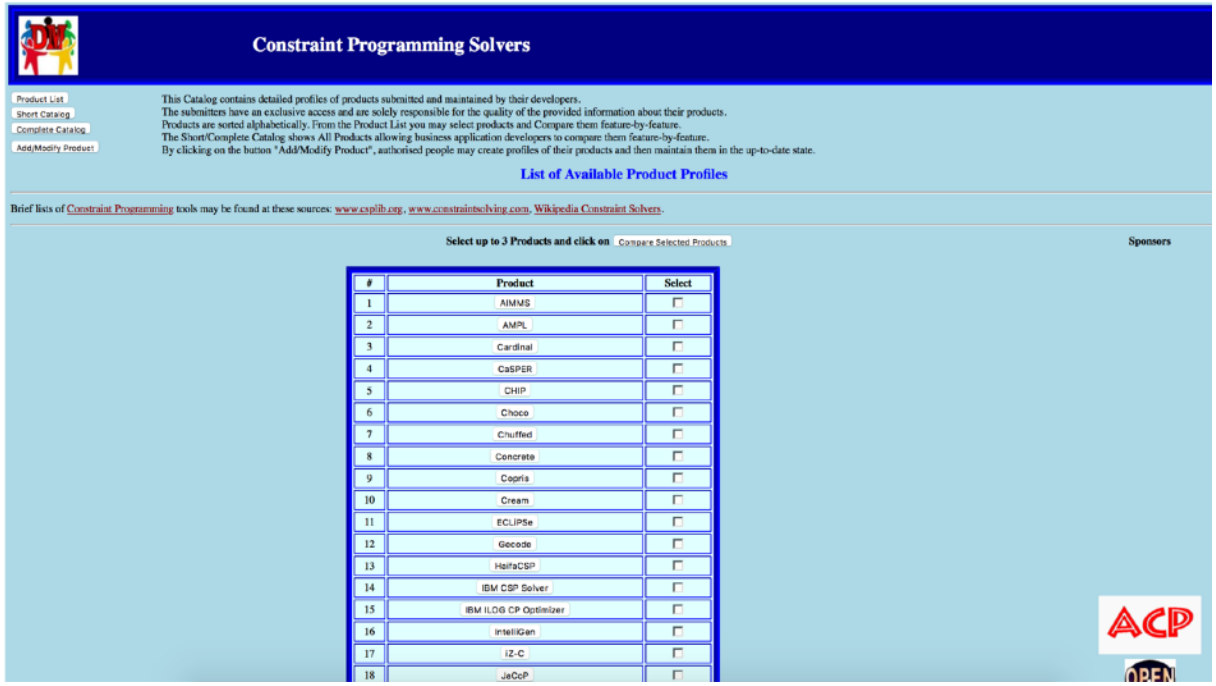
CP solver: A library of constraints → **Model + Resolution**

CP Modelling = Putting together the different constraints of the problem.

CP Solvers Catalog

Many CP-solvers exist to ease the implementation of model.

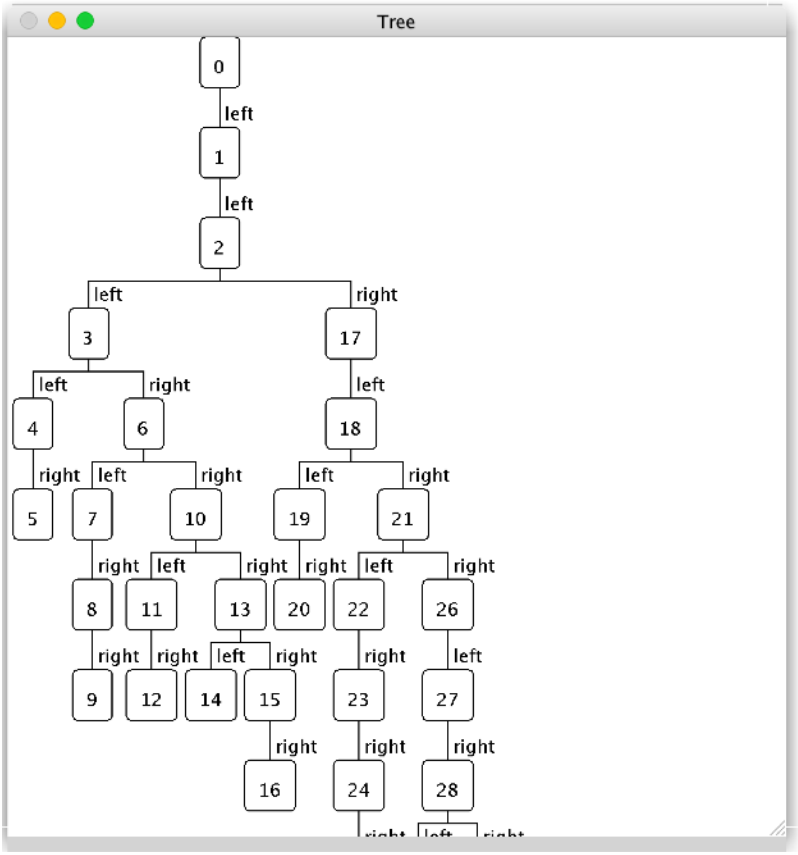
http://openjvm.jvmhost.net/CPSolvers/

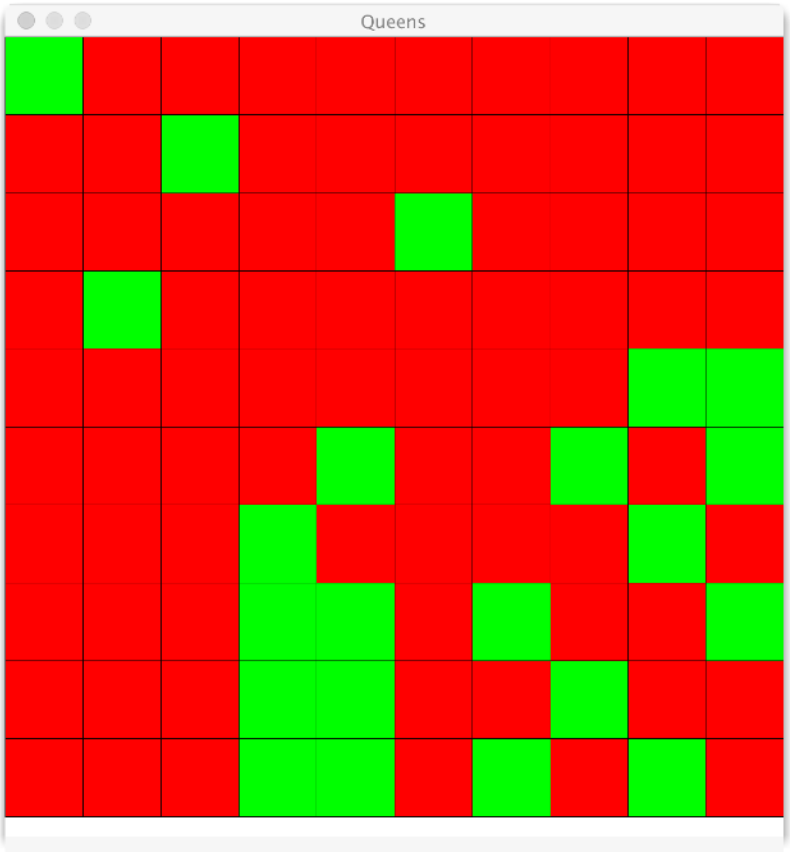


This catalog presents a list of 35 CP based solvers (August 13, 2018).

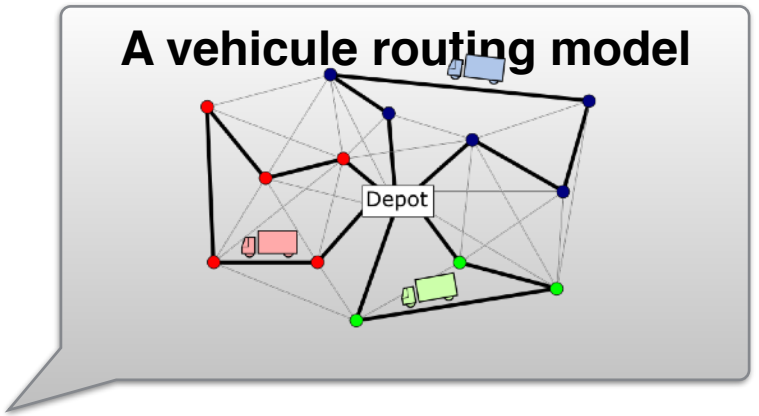
How CP works concretely?

A live demo with the solver OscaR.





How CP works concretely?



Put together the different constraints.

The job of a constraint !

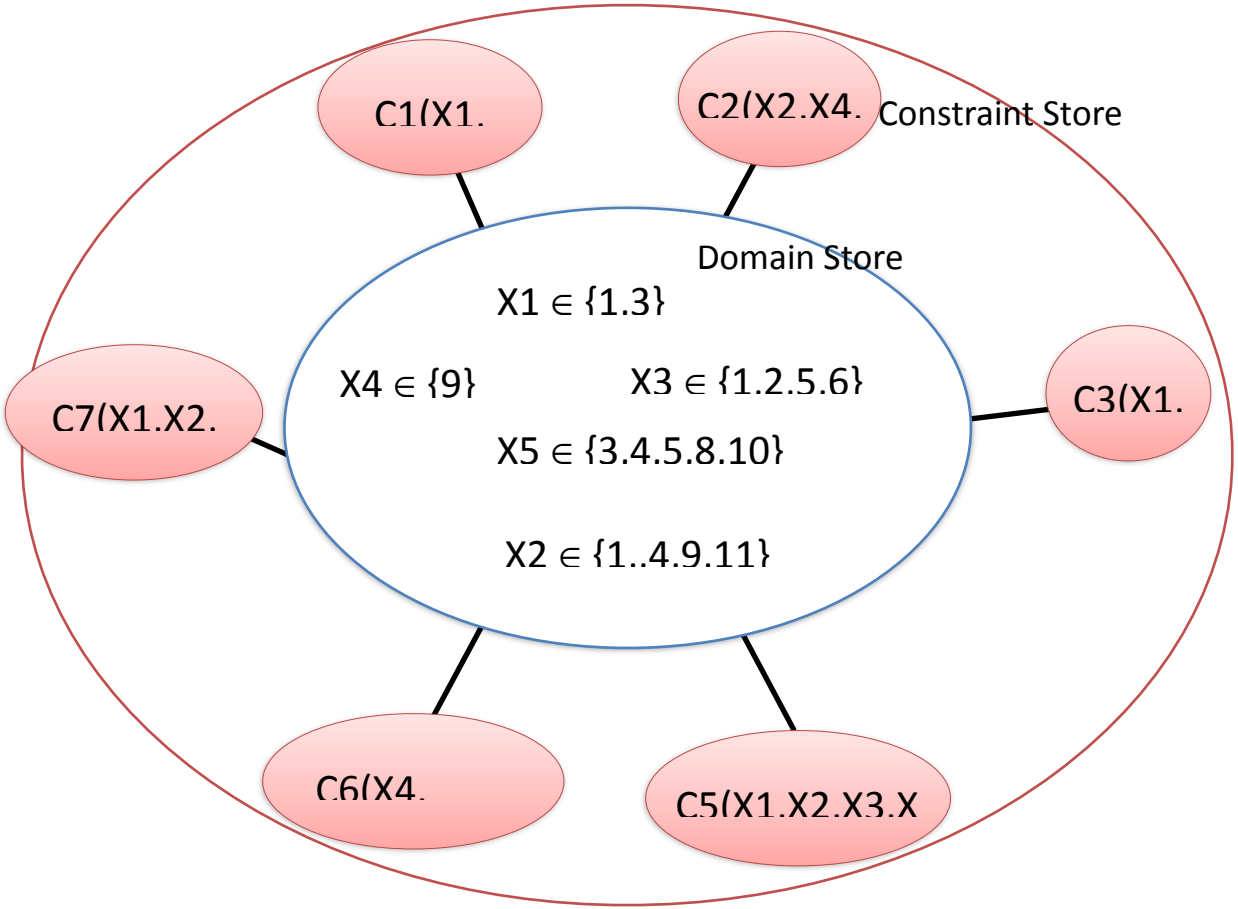
- Given the status of the domain, do you think that you still have a solution?

- Remove the values from the domains that you think are impossible in a solution (= **filtering or pruning**)

How CP works concretely?

Each constraint must remove impossible values.

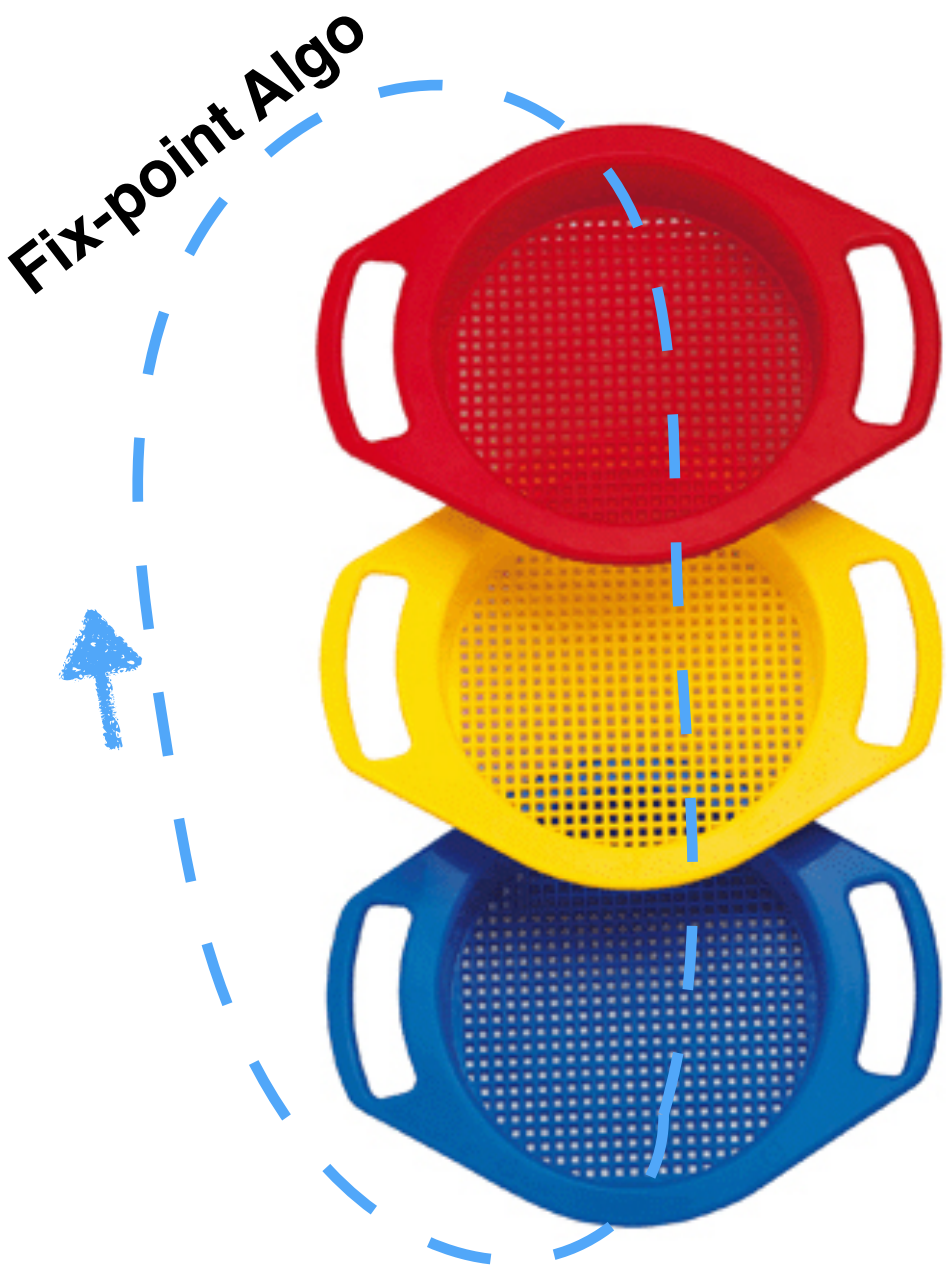
Yes but the pruning of one constraint might trigger new deductions for other constraints…



Fix-Point Algorithm

```
repeat
   select a constraint c
   if c is OK wrt the domain store
     apply filtering algorithm of c
   else
      return KO
until no value can be removed
```

Back to our gold seeker illustration

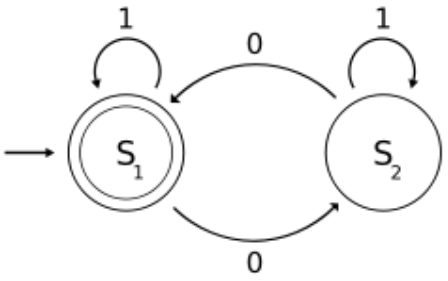


sum(X[]): $\sum_i X(i) = 0$

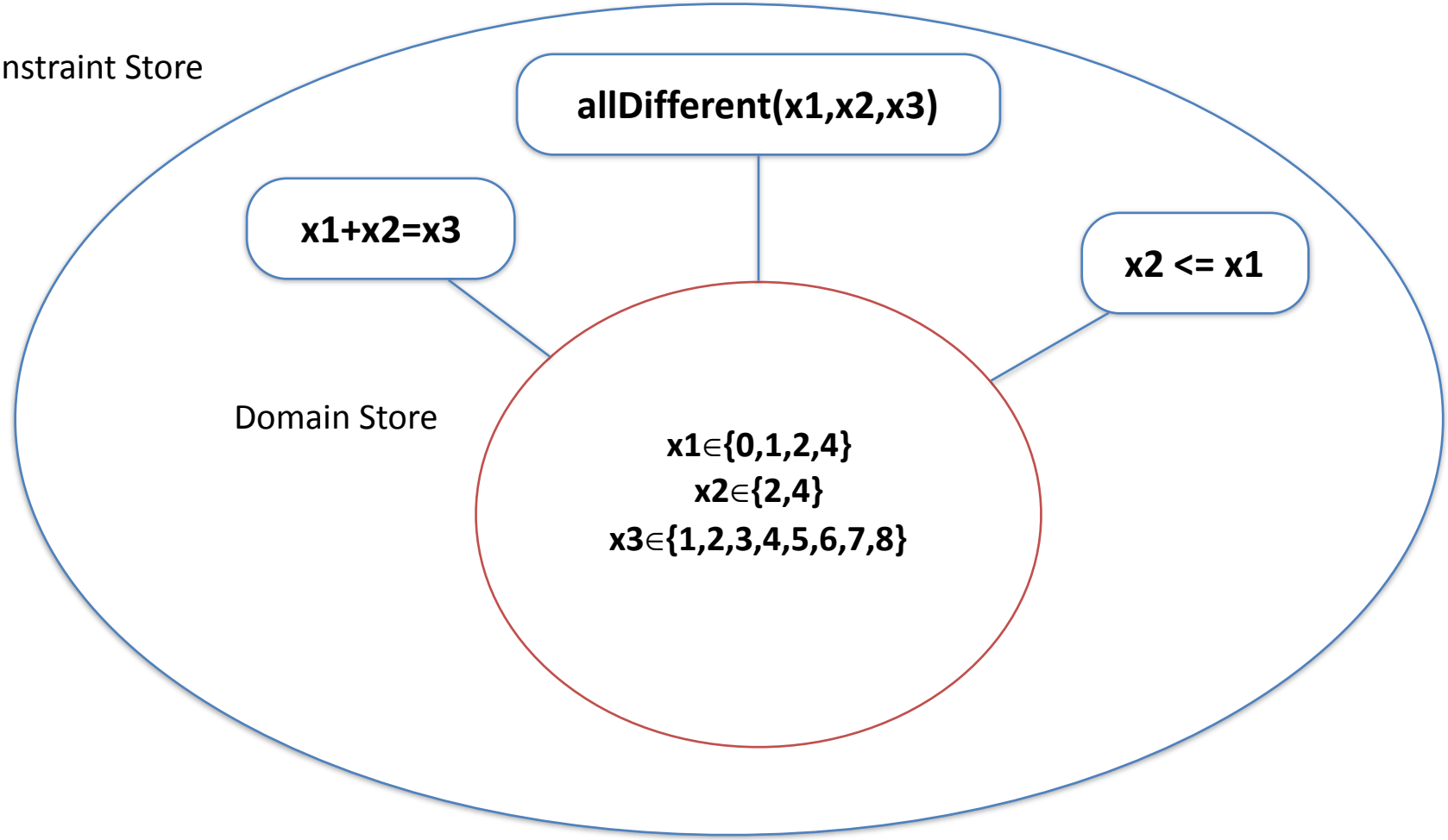
allDifferent(X[]):

$\forall i < j : X(i) \neq X(j)$

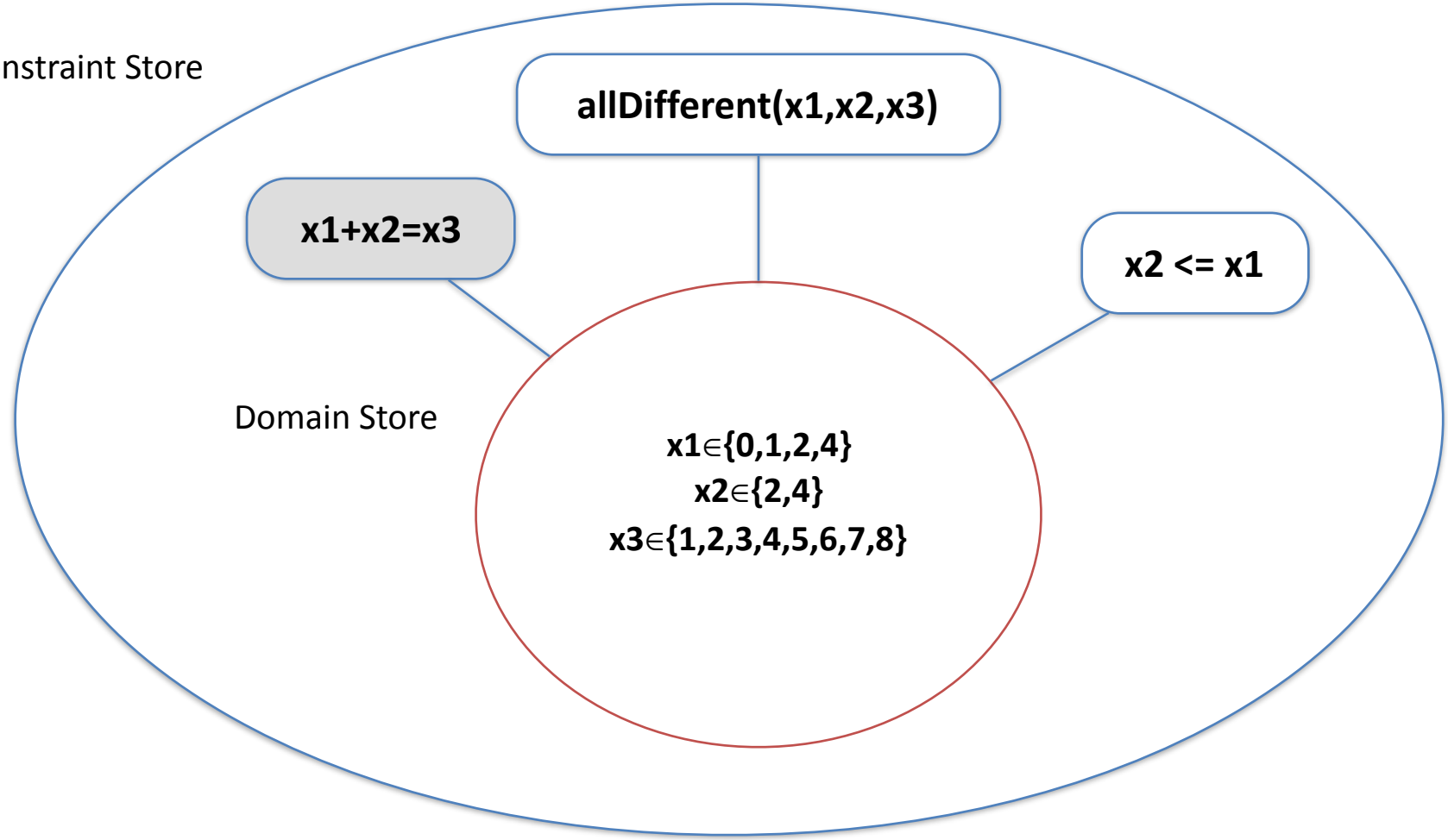
regular(X[]):



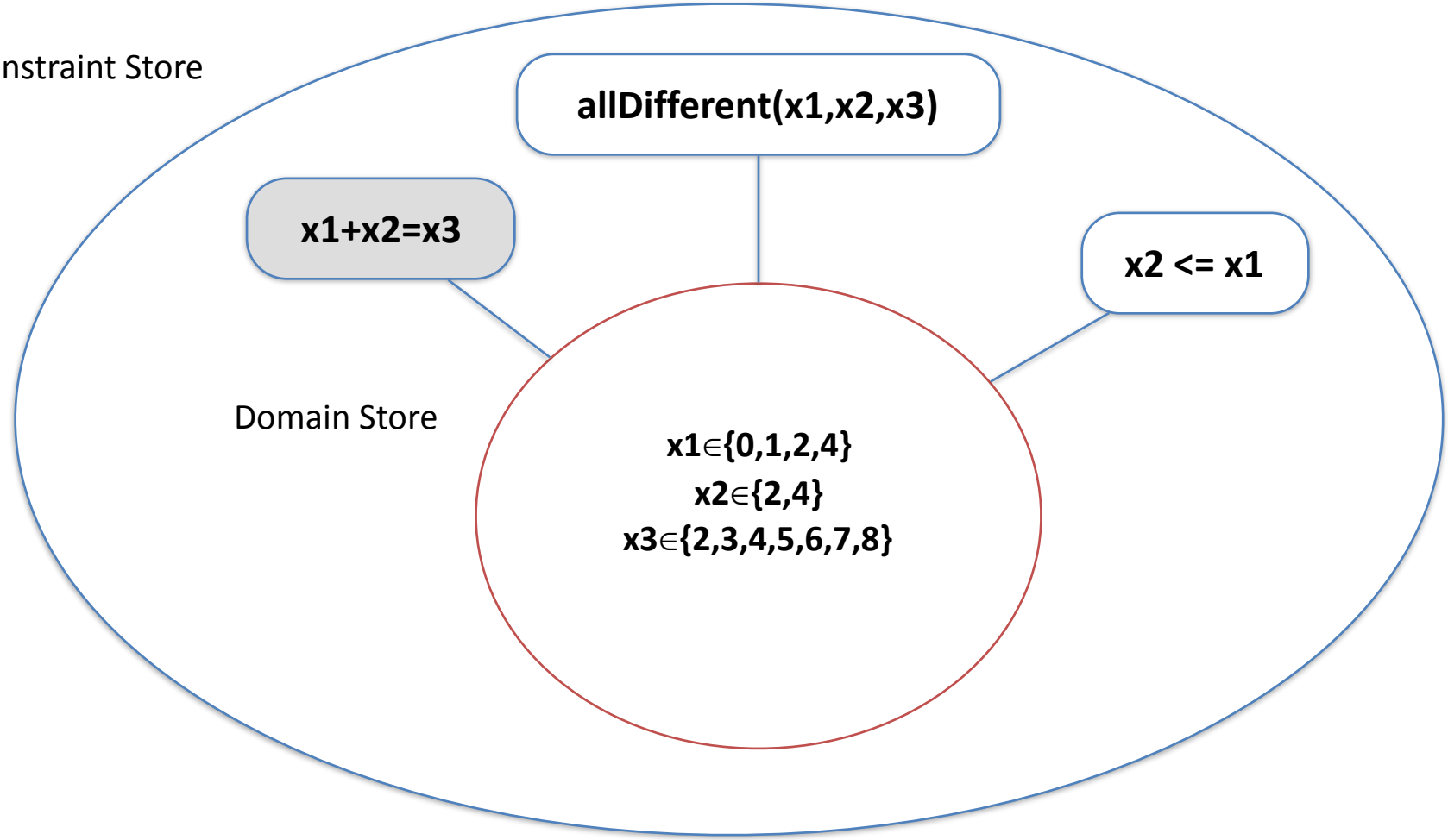
Fix-Point Example



How CP works concretely?

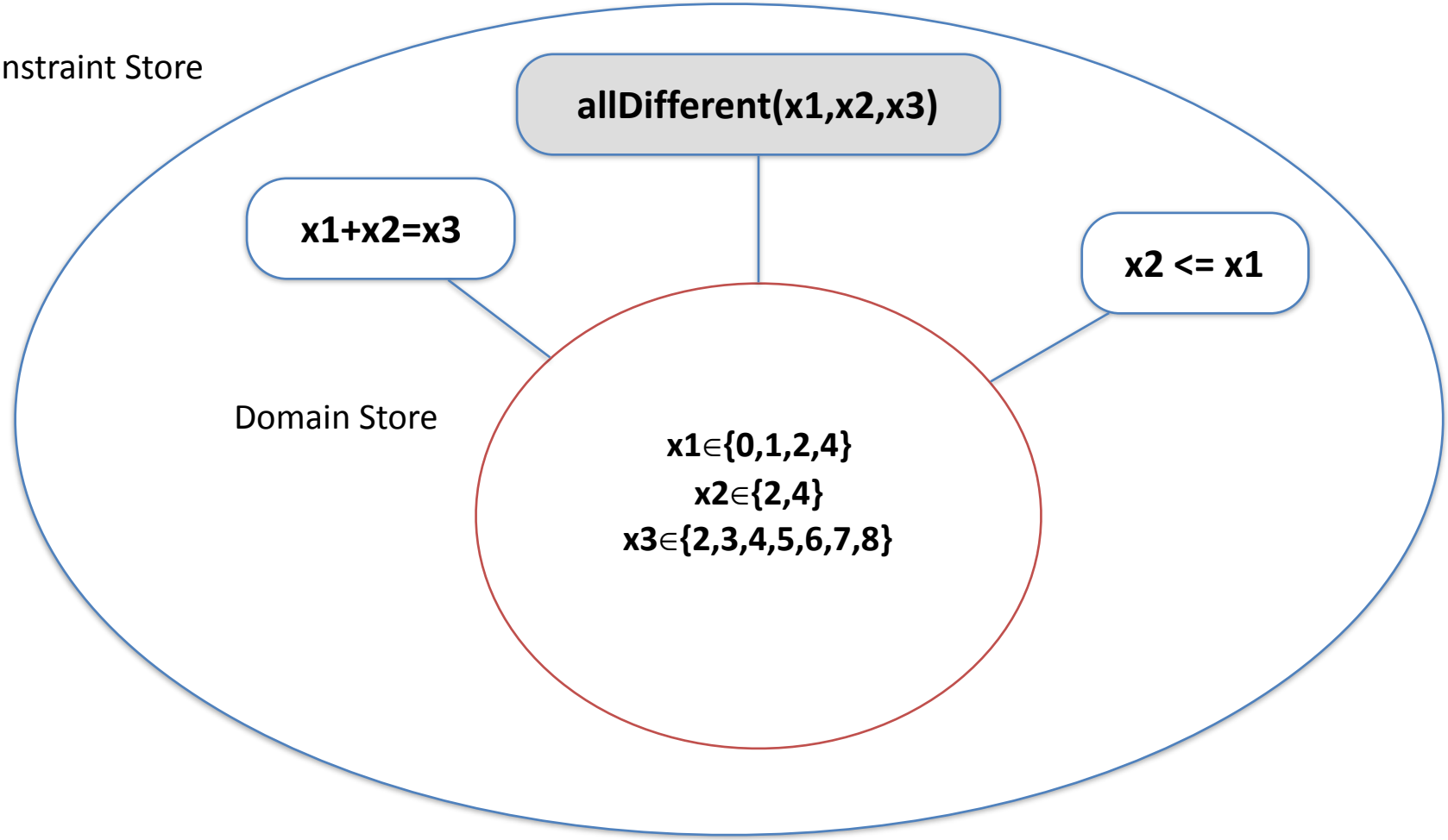


How CP works concretely?



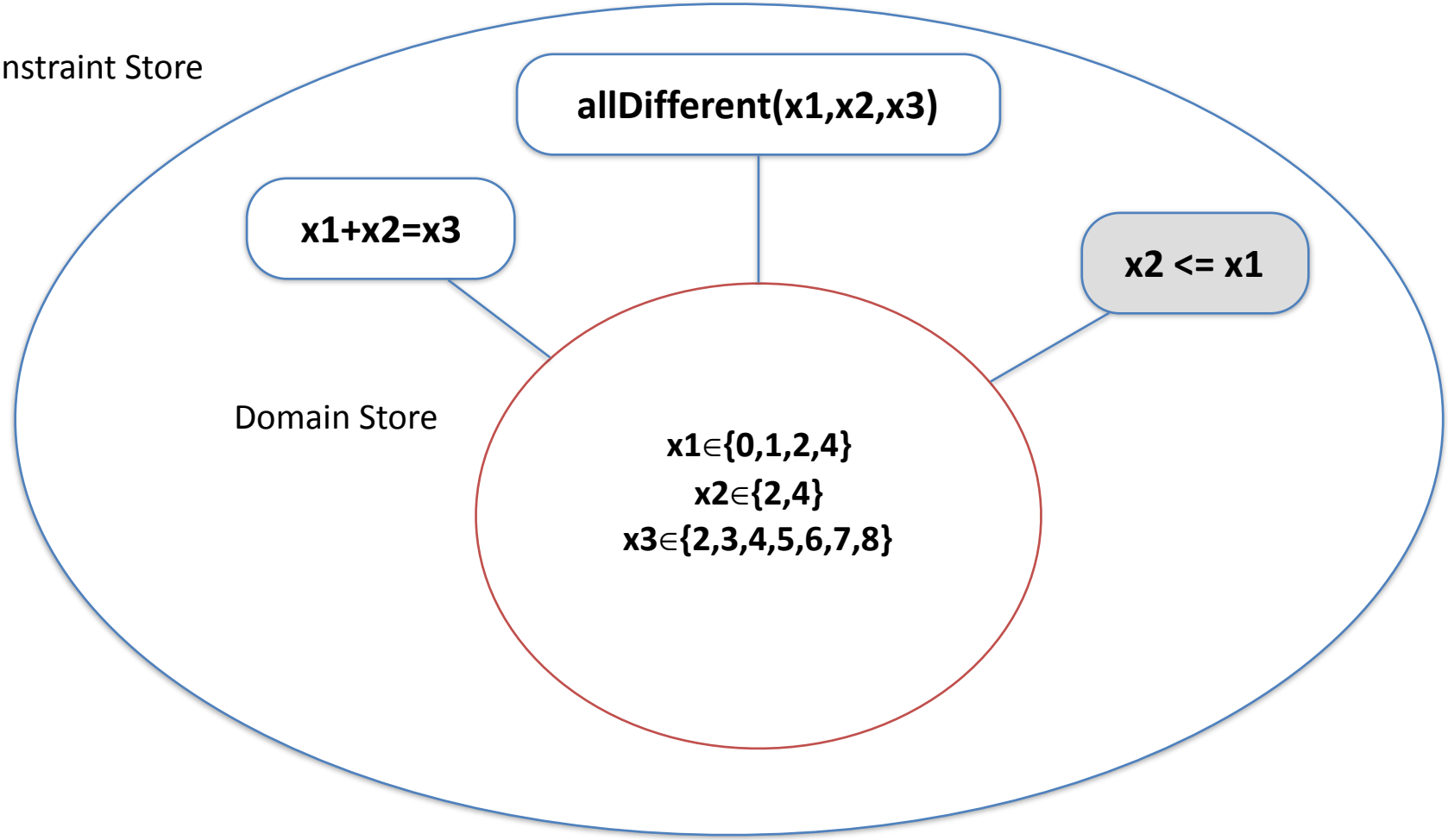
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



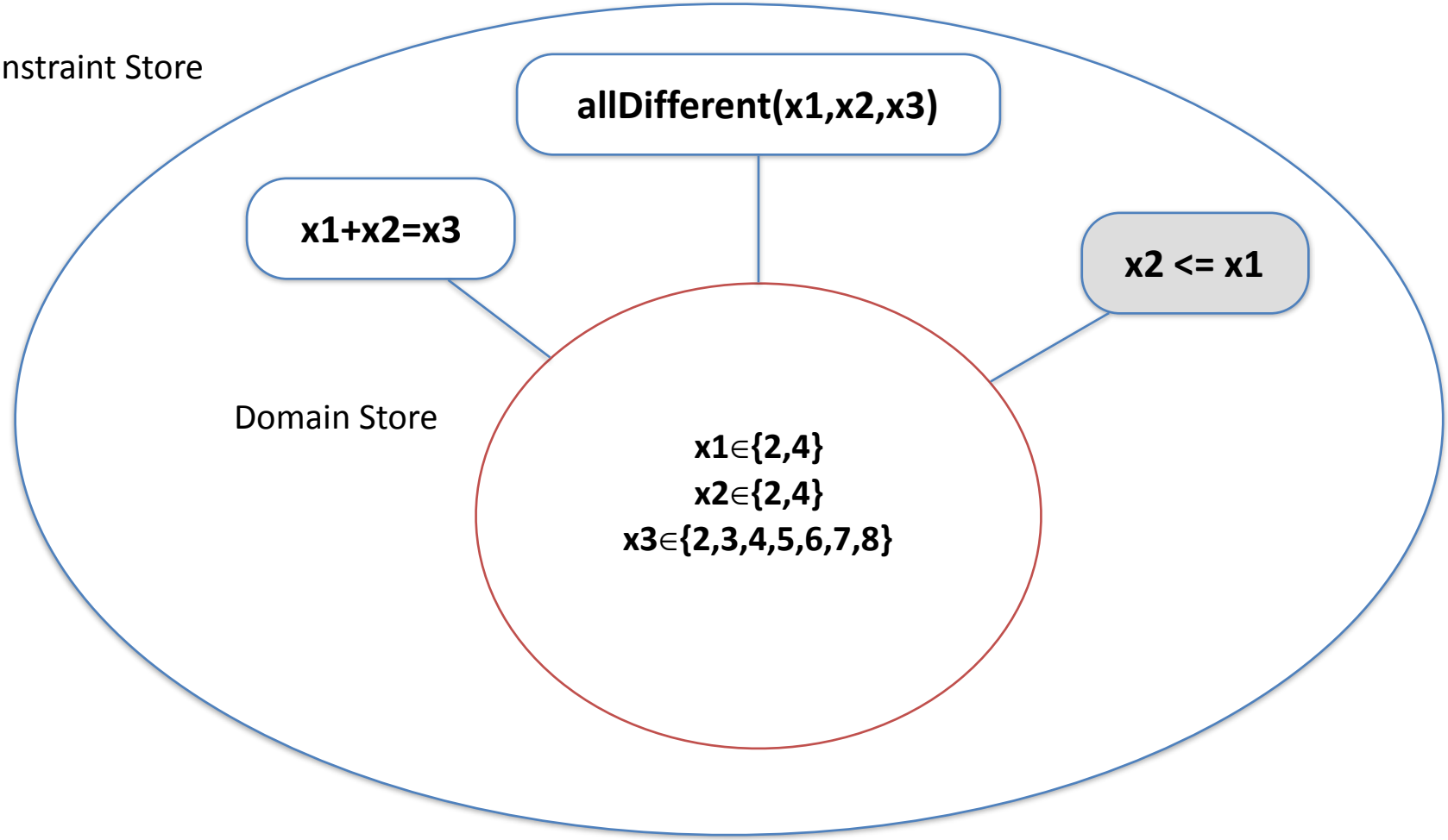
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



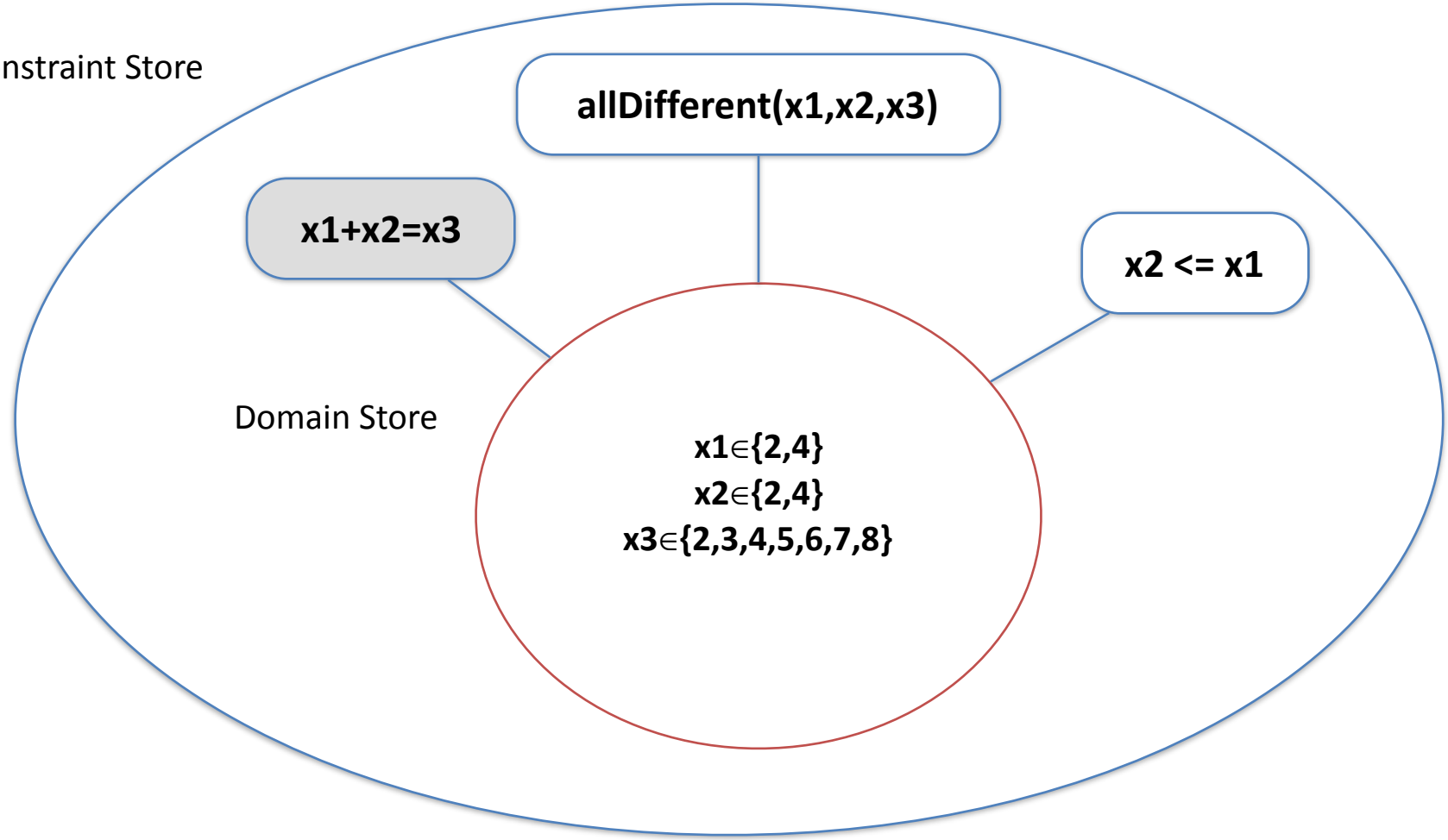
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



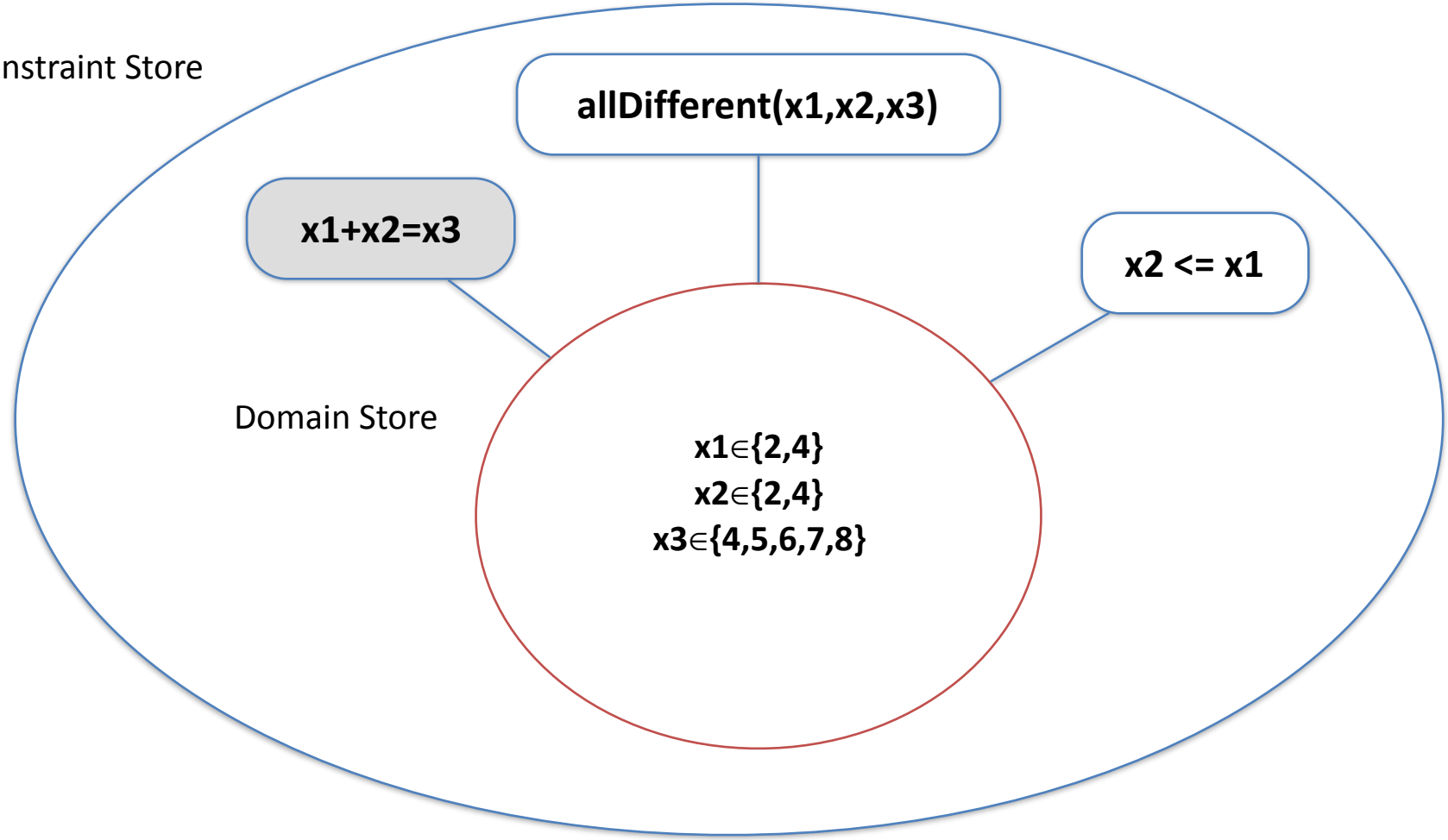
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



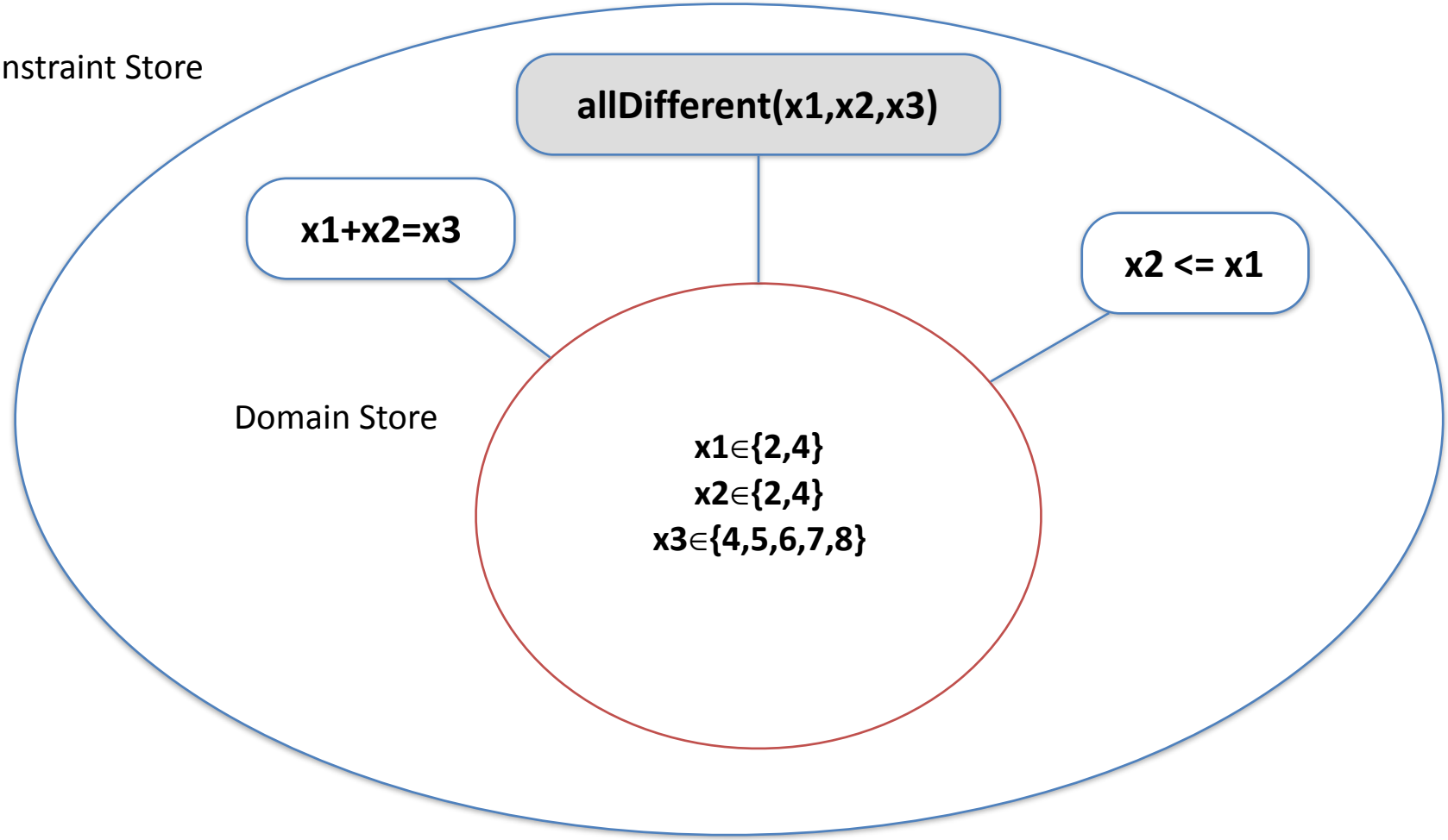
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



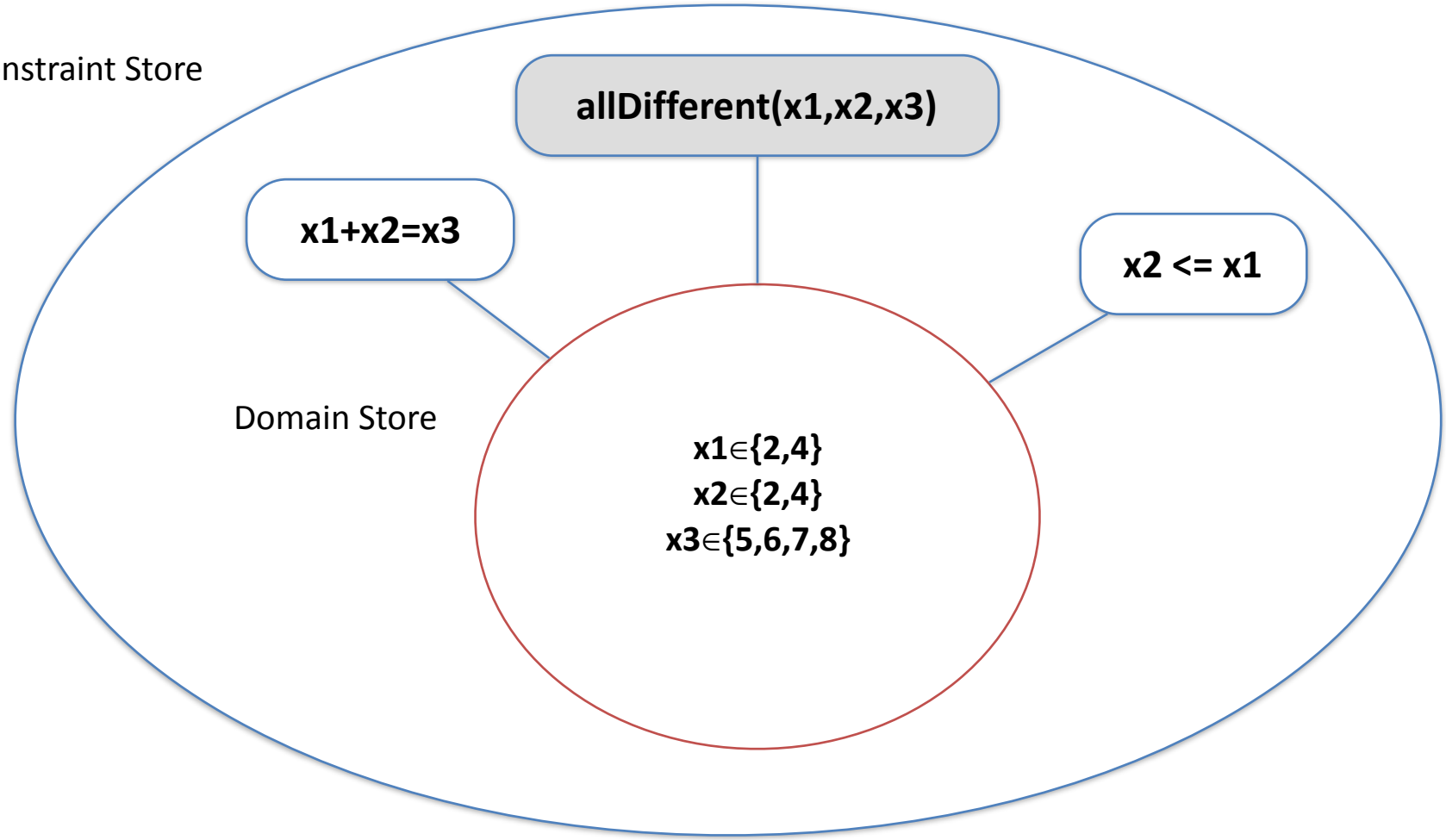
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



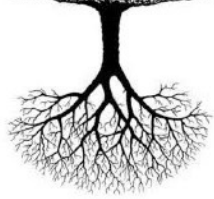
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



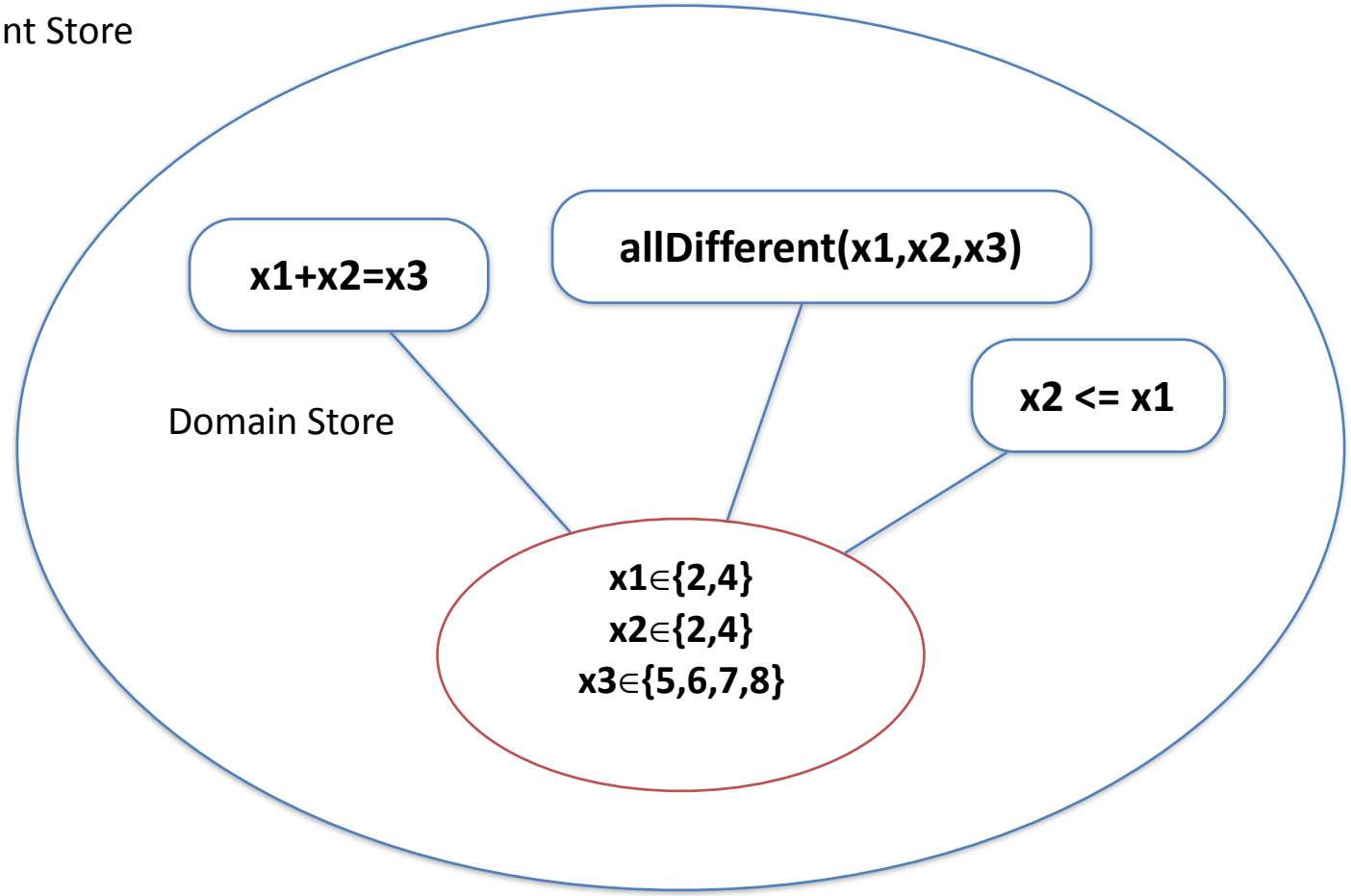
ps: We assume the sum constraint only reasons on bounds of the domains

How CP works concretely?



Fix-Point! but not a solution yet …

We need to search …

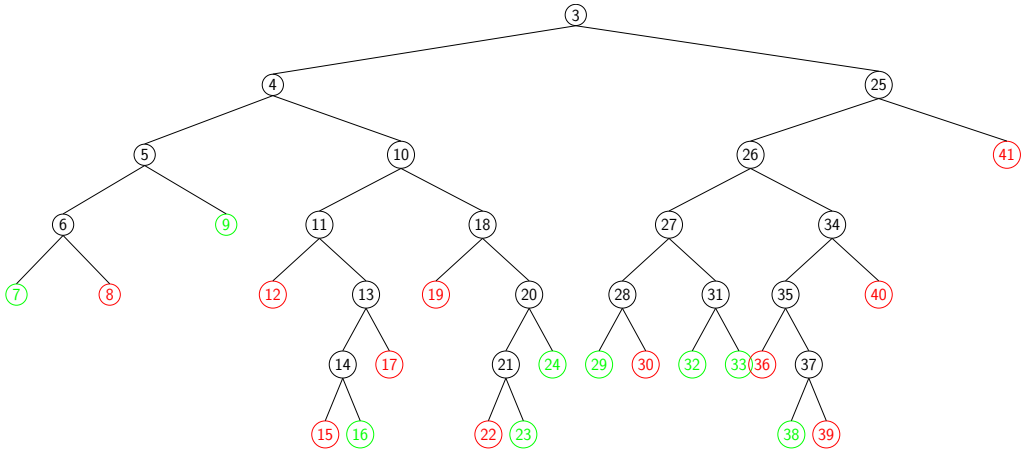


ps: We assume the sum constraint only reasons on bounds of the domains

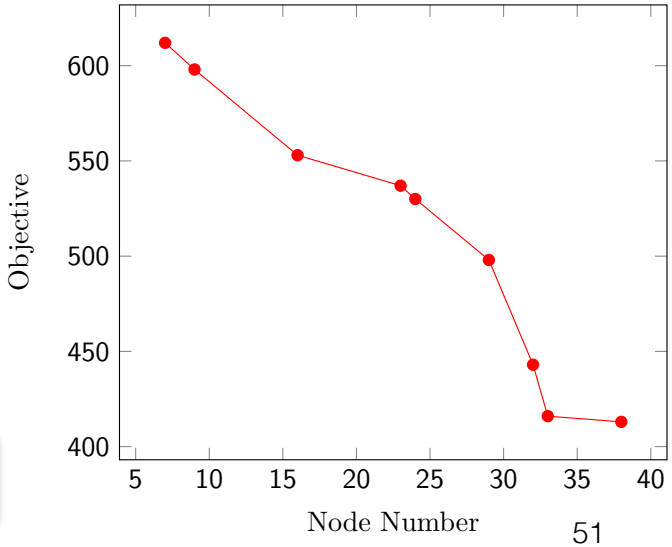
Optimization with CP

```
  // your constraints
cp.minimize(objVar)
} search {
   branching(decVars)
} start()
```

```
// your constraints
} search {
   branching(decVars)
} onSolution {
  updateBound()
} start()
```



Best solution



Outline

General Introduction

Global Constraints

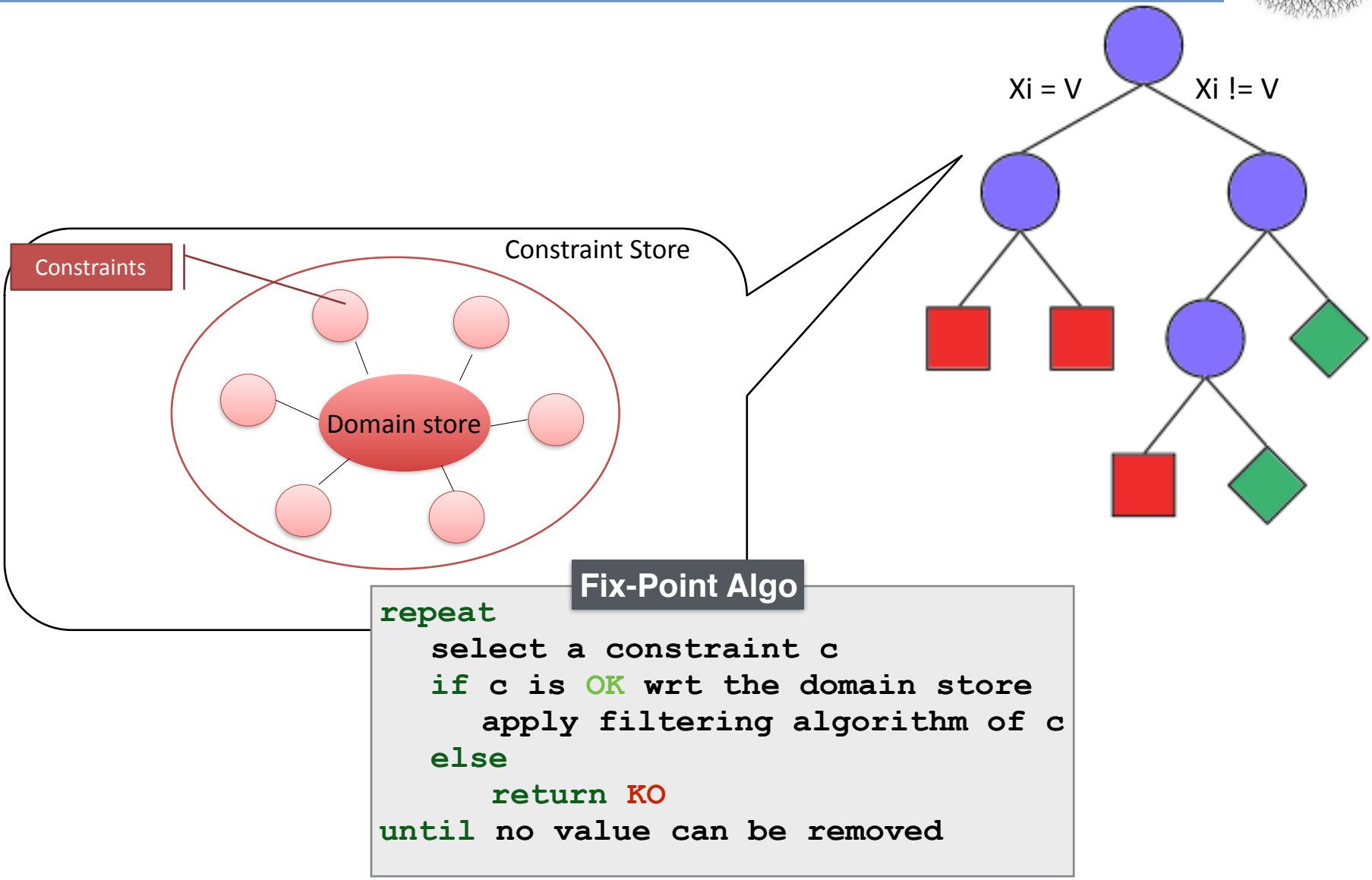
How CP works concretely ?

Take Away Message

Take away message

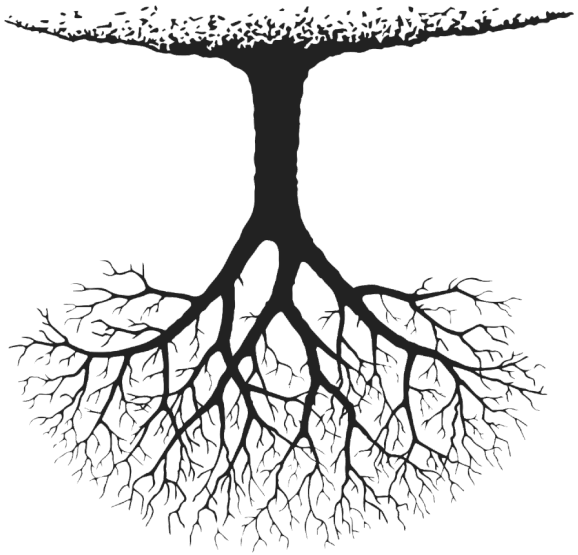
You must experiment CP !!

Take away message



Fix-Point Algo

```
repeat
   select a constraint c
   if c is OK wrt the domain store
      apply filtering algorithm of c
   else
      return KO
until no value can be removed
```



Thank you for your attention

Some references

Pierre Schaus, Constraint Programming: A tool inspired by gold seekers. Grascomp, 2015. [Some slides are from this reference]

CSPLib: A problem library for constraints. csplib.org

A catalog of global constraints. http://sofdem.github.io/gccat/

CP Solvers Catalog. http://openjvm.jvmhost.net/CPSolvers/

Mini-CP: A lightweight Constraint Programming Solver http://www.minicp.org/

OscaR is a Scala toolkit for solving Operations Research problems. https://bitbucket.org/oscarlib/oscar/wiki/Home