RULES FEST

Constraint Programming 101

Jacob Feldman, Ph.D.

Founder & CTO

OpenRules Inc.

“Shock Troops” for Enterprise Decision Management

“I have concluded that decision making and the techniques and technologies to support and automate it will be the next competitive battleground for organizations. Those who are using business rules, data mining, analytics and optimization today are the shock troops of this next wave of business innovation”

Tom Davenport

Business Optimization

- Optimization usually refers to a mathematical technique used to calculate the *best* possible resource utilization to achieve a desired optimization objective such as:
  - minimizing expenses or travel time
  - maximizing ROI or service level, etc.
- Business Optimization helps *business* people to find optimal solutions among multiple alternatives subject to different *business* constraints
- Optimization Engine:
  - Determines how to most effectively allocate resources, automatically balancing trade-offs and business constraints
  - Eliminates the need to manually work out plans and schedules, so you can achieve maximum operational efficiency

Constraint Programming (CP)

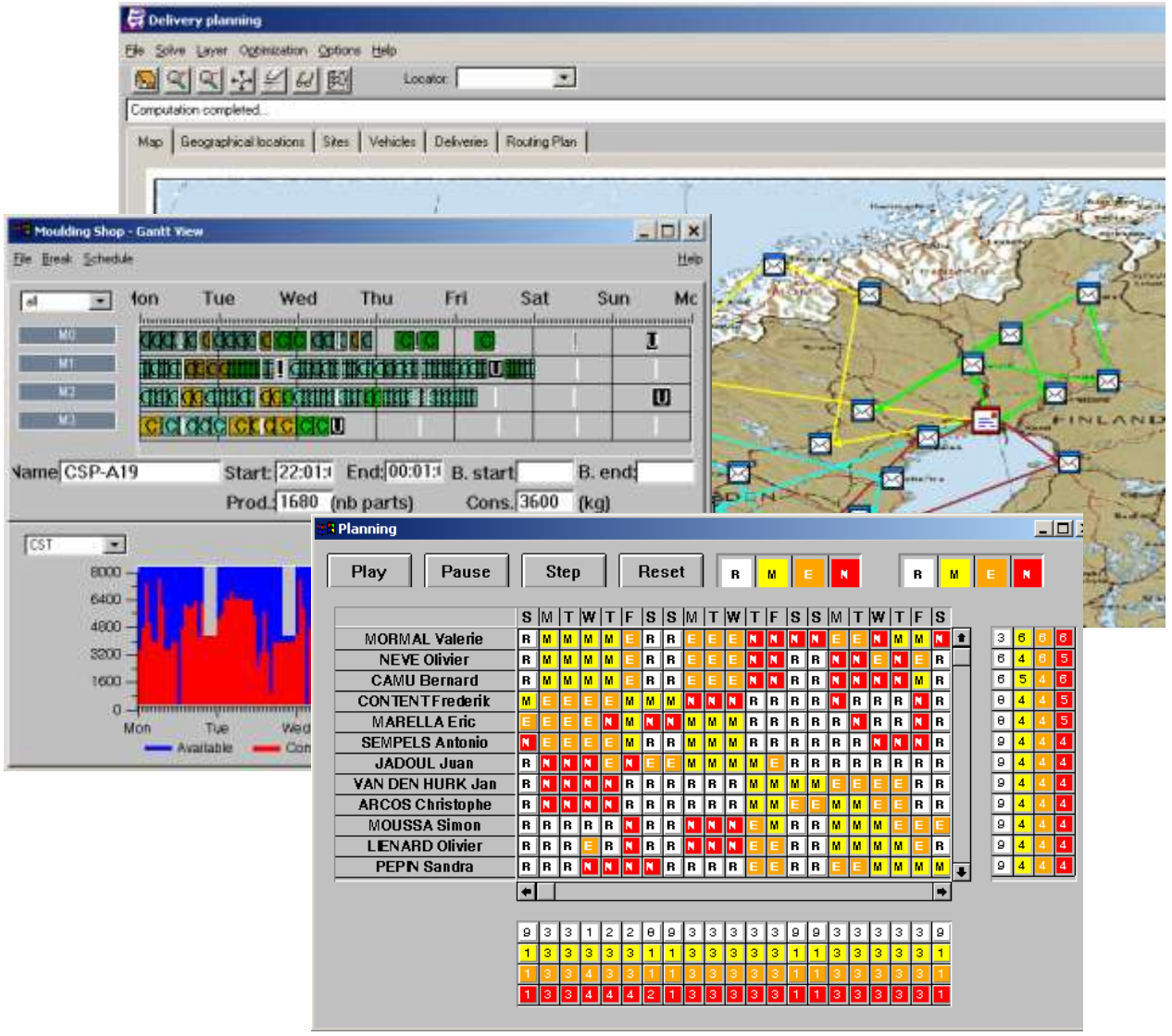
- Constraint Programming (CP) is a proven *optimization* technology introduced to the business application development at the beginning of 1990s
- Constraint Programming is a very powerful problem solving paradigm with strong roots in Operation Research and AI:
  - *Handbook of Constraint Programming* (Elsevier, 2006)
  - ACP - Association for Constraint Programming - http://slash.math.unipd.it/acp/
  - Cork Constraint Computation Centre - http://www.4c.ucc.ie/
- CP was especially successful dealing with real-world scheduling, resource allocation, and complex configuration problems:
  - CP clearly separates problem definition from problem resolution bringing declarative programming to the real-world
  - CP made different optimization techniques handily available to software developers (without PhDs in Operation Research)

Constraint Programming: a bridge between academy and business

- During the 90s CP successfully built a bridge between the academic and business worlds by providing an API for the mainstream programming languages
- Constraints arise in design and configuration, planning and scheduling, diagnosis and testing, and in many other contexts
- CP was successfully applied to solve real-world problems in:
  - telecommunications, internet commerce, electronics, bioinformatics, transportation, network management, supply chain management, finance, manufacturing, and many other fields

Typical CP Applications

- Scheduling and Resource Allocation
- Complex Configuration Problems
- Supply Chain Management
- Staff Rostering
- Vehicle Routing



Procedural vs Declarative Programming

- **Simple example of a constraint satisfaction problem:**

There are three integers **x**, **y**, and **z** defined from **0** to **10**.
Our goal is to find the solution that would maximize or minimize the objective function represented by the following integer expression:
cost=3x*y-4*z

subjected to:
x < y
x + y = z

  - Write A Pure Java solution
  - Consider a CP-based solution

CP-Based Solution

```
public class Test {

    public static void main(String[] args) {
        // ==== PROBLEM DEFINITION ==============================
        Problem p = ProblemFactory.newProblem("Test");
        // ======= Define variables
        Var x = p.variable("X", 1, 10);
        Var y = p.variable("Y", 1, 10);
        Var z = p.variable("Z", 1, 10);
        Var cost = x.multiply(3).multiply(y).minus(z.multiply(4));
        // ======= Define and post constraints
        p.post(x, "<", y); // X < Y
        p.post(x.plus(y), "=", z); // X + Y = Z

        // === PROBLEM RESOLUTION ================================
        p.log("=== Find Solution:");
        Solver solver = p.getSolver();
        Solution solution = solver.findSolution();
        if (solution != null)
            solution.log();
        else
            p.log("No Solution");
        p.log("Cost " + cost);
    }
}
```

Constraint Satisfaction Problem - CSP

- Typical CSP structure:
  1. **Problem Definition (what to do)**
     a. Define Constrained Variables with all possible values
     b. Define Constraints on the variables
  2. **Problem Resolution (how to do it)**
     a. Find Solution(s) that defines a value for each variable such that all constraints are satisfied or
     b. Find an optimal solution that minimizes/maximizes a certain objective

How the constraint “X < Y” works

- Let’s assume X and Y are defined on the domain [0,10]
- Initial constraint propagation after posting X<Y constraint:

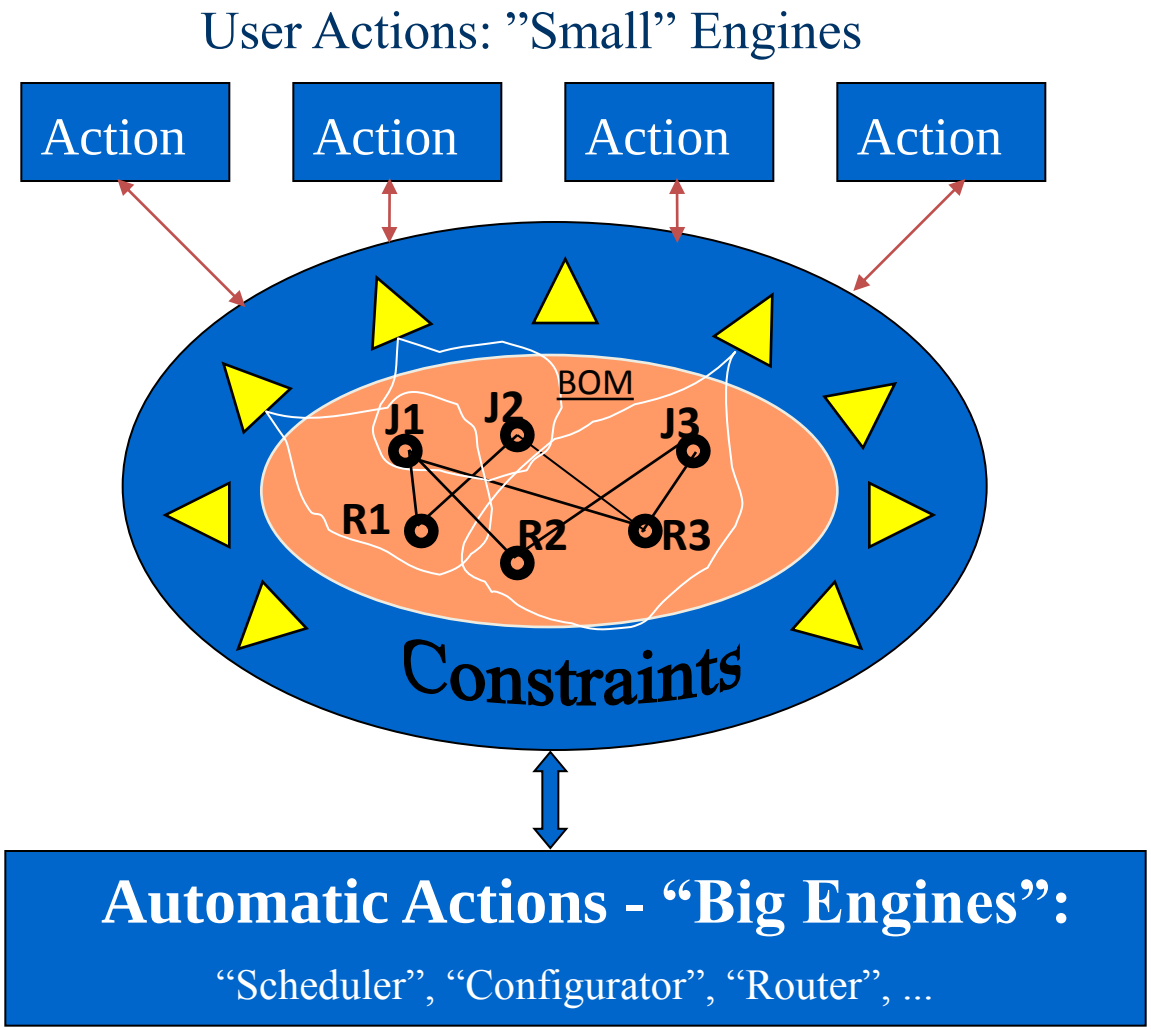
  X[0;9]

  Y[1;10]
- Changes in X cause the changes in Y

  X>3 → Y > 4
- Changes in Y cause the changes in X

  Y<=8 → X<=7
- Bi-Directional constraint propagation

Constraint Propagation (intuitive view)



Demo “CP Online”

Constraint Satisfaction Environment

- Predefined classes for Constrained Objects, Constraints, and Search Goals
- Domain representations for major constrained objects
- Generic reversible environment
  - “Try/Fail/Backtrack” capabilities
  - Powerful customizable event management mechanism
  - Constraints use events to control states of all constrained objects
- Constraint propagation mechanisms
- Ability to write problem-specific constraints and search goals
- Typical Solver Implementations:

C++ , Java, Prolog, different CP Modeling Languages

JSR-331 – Java Specification Request

- Java Constraint Programming API under the Java Community Process (JCP) www.jcp.org
- JSR-331 covers key concepts and design decisions related to the standard representation and resolution of constraint satisfaction and optimization problems
- Utilizes de-facto standardized decisions from multiple CP solvers

Some Popular CP Tools

- Java API: JSR-331
  - **Choco**, **Constrainer, JaCoP**
- C++ API
  - **ILOG CP** – Commercial (www.ilog.com)
  - **Gecode** – Open Source (www.gecode.org)
- CP environments with specialized modeling languages
  - **OPL** from ILOG, France (www.ilog.com)
  - **MiniZinc** from G12 group, Australia (http://www.g12.cs.mu.oz.au)
  - **Comet**, Brown University (www.comet-online.org)
  - **Prolog-based tools** (ECLiPSe, SICStus)
- 20+ other CP Solvers: http://slash.math.unipd.it/cp/
- CP Solvers are usually well integrated with other optimization tools (LP, MIP)

JSR-331 CP API: Basic Concepts

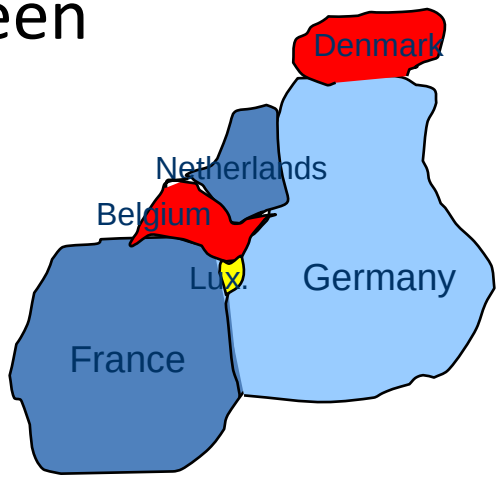
- Problem – the main class that defines and solves constraint satisfaction problems. A placeholder for all other objects and methods
- Var – defines constraint integer variables
- VarReal – defines constraint real variables
- VarBool – defines constraint real variables
- VarSet – defines constraint set variables
- Constraint – defines various constraints
- Solver
- Search Strategy

CP API: Examples

- Analyse basic JSR-331 arithmetic examples:
  - XYZ1 – find a solution
  - XYZ2 – find all solutions
  - XYZ3 – find an optimal solution

CSP Sample: “Map Coloring”

- A map-coloring problem involves choosing colors for the countries on a map in such a way that at most 4 colors are used and no two neighboring countries have the same color
- We will consider six countries: Belgium, Denmark, France, Germany, Netherlands, and Luxembourg
- The colors are blue, white, red or green

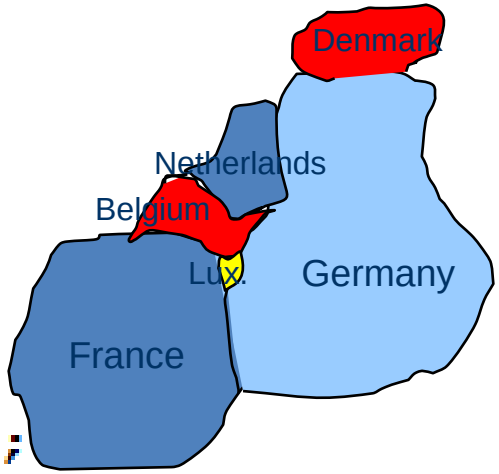


Example “Map Coloring”: problem variables

```
Problem  p = ProblemFactory.newProblem("MapColoring");
// Define Variables
Var Belgium      = p.variable("Belgium“,0, 3);
Var Denmark      = p. variable("Denmark“.0, 3);
Var France       = p.variable("France“,0, 3);
Var Germany      = p.variable("Germany“,0, 3);
Var Netherlands = p.variable("Netherlands“,0, 3);
Var Luxemburg   = p.variable("Luxemburg“,0, 3);
```

Each country is represented as a variable that corresponds to an unknown color: 0,1,2, or 3

“Map Coloring”: problem constraints



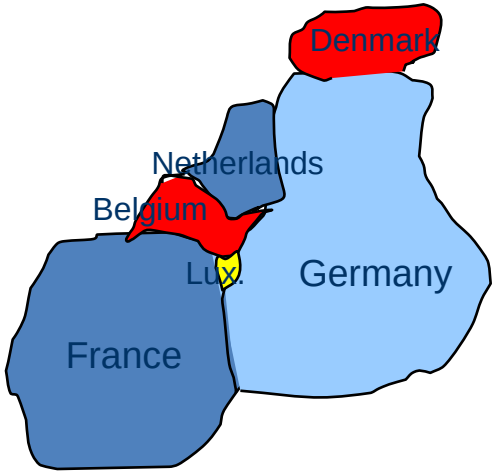

```
p.post(France, "!=", Belgium);
p.post(France, "!=", Luxemburg);
p.post(France, "!=", Germany);
p.post(Luxemburg, "!=", Germany);
p.post(Luxemburg, "!=", Belgium);
p.post(Belgium, "!=", Netherlands);
p.post(Belgium, "!=", Germany);
p.post(Germany, "!=", Netherlands);
p.post(Germany, "!=", Denmark);
```

“Map Coloring”: solution search

```
Solution solution = p.getSolver().findSolution();
if (solution != null) {
    solution.log();
    for (int i = 0; i < p.getVars().length; i++) {
        Var var = p.getVars()[i];
        p.log(var.getName() + " - "
                + colors[solution.getValue(var.getName())]);
    }
} else
    p.log("no solution found");
```

// Solution:

Belgium – red
Denmark – red
France – green
Germany – blue
Netherlands – green
Luxemburg - yellow



Lab: Solve Logical Puzzle

SEND + MORE + MONEY

This example shows how to represent and solve a simple puzzle:

```
  S E N D
+ M O R E
-----------
M O N E Y
```

where different letters represent different digits.

Lab: solve a logical problem: “Zebra”

The following is a quiz which rumors say was written by Einstein.
He stated that as much as 98% people of the world can not solve this quiz.
So, prove yourself that you are included in the rest 2% who not only can
solve this quiz, but also do it only under 15 minutes.

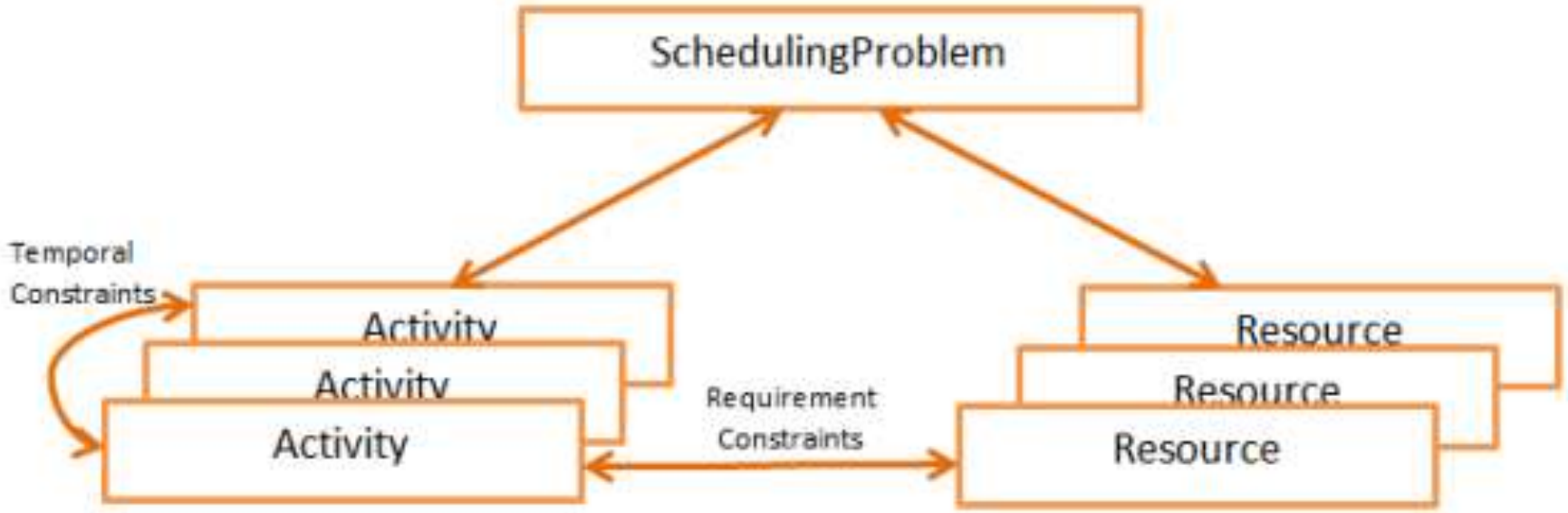
Here are the facts:
(1) There are 5 houses, each painted with a different color.
The house is numbered from 1 to 5, from left to right.
(2) Each house is occupied by only one person; each person has
a different nationality from the others.
(3) Each person drinks a different drink, smokes a different cigarette,
and has a different pet.

Now, here are the known constraints:
(1) The British lives in the red house.
(2) The Spaniard has a dog.
(3) The owner of the green house drinks coffee.
(4) The Ukrainian drinks tea.
(5) The green house is located at the right side of the white house.
(6) The person who smokes OldGolds has a snail.
(7) The owner of the yellow house smokes Kools.
(8) The person, who lives in the house exactly in the middle, drinks milk.
(9) The Norwegian lives in the house numbered one.
(10) The person who smokes Chesterfield lives next to the person who has a fox.
(11) The person who has a horse lives next to the person who smokes Kools.
(12) The person who smokes LuckyStrike drinks juice.
(13) The Japanese smokes Parliament.
(14) The Norwegian lives next to the the blue house.

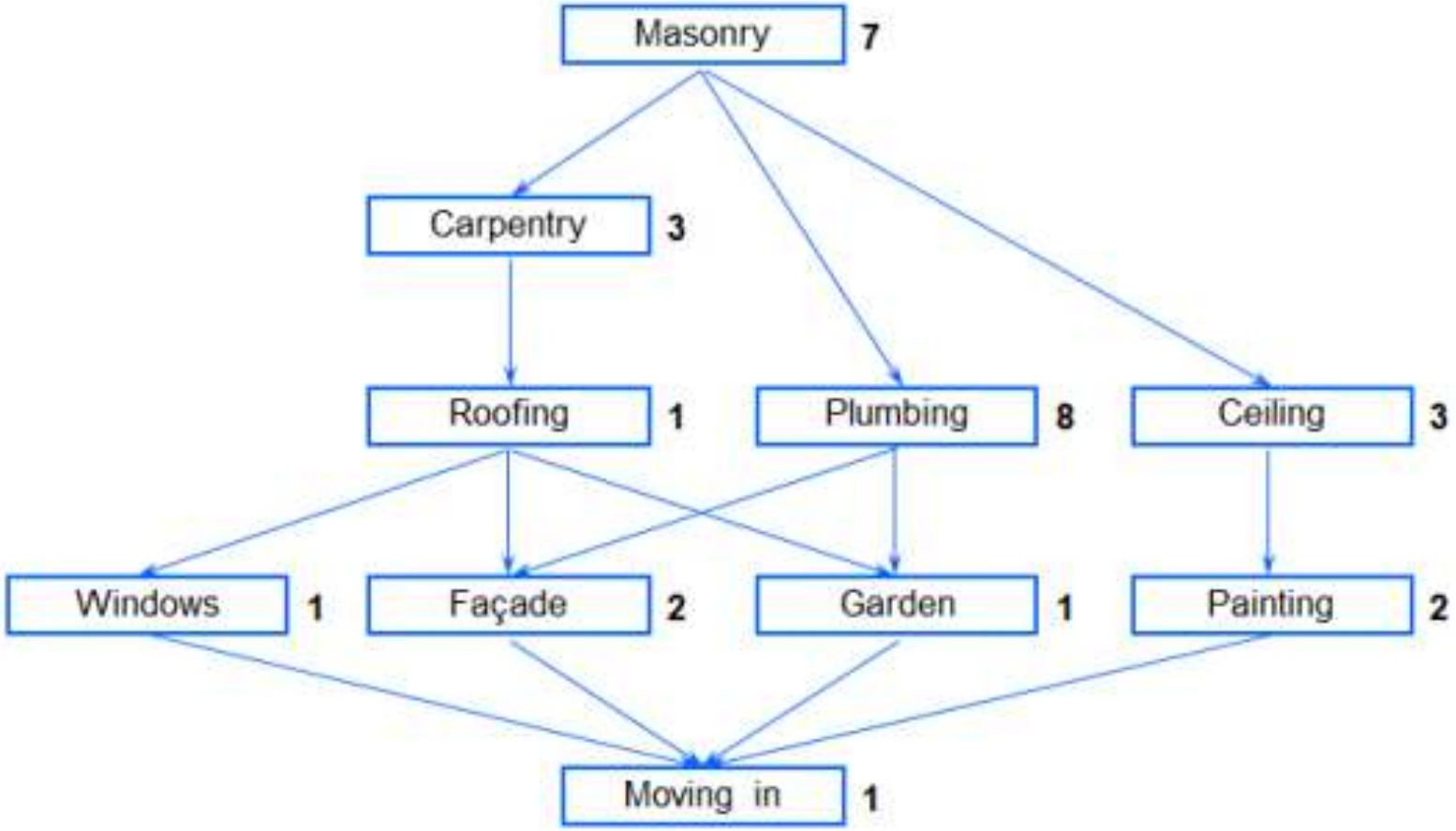
The question is:
Who has a ZEBRA?

Let’s apply CP to solve the problem!

Scheduling Problems

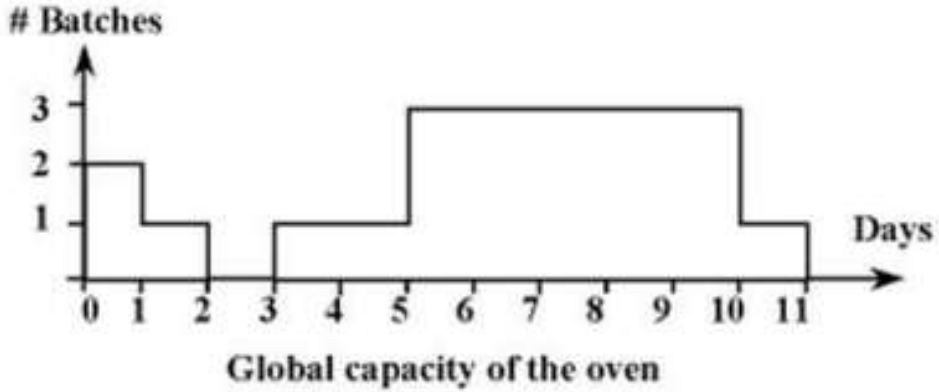


House Construction Problem



Resource Allocation Problem

The following problem deals with activities that require a common resource. Let's consider 5 different orders (activities) that fire batches of bricks in an oven (a resource with a limited capacity). Each order 's size and duration, as well as the oven's capacity, are described in the following figure:



A 2 batches, 1 day

B 1 batch, 4 days

C 1 batch, 4 days

D 1 batch, 2 days

E 2 batches, 4 days

5 Activities

Q&A