# LINGI2365 : Constraint Programming

## Assignment 1 :

Introductory Problems

F.Aubry & Y. Deville

*February 2014*

# 1 Objectives and practical details

The goal of this assignment is to introduce you to a constraint programming system. You will be asked to model simple problems and implement the models in Comet. The main objective is that you get accustomed with Comet as it will be used for the subsequent assignments.

### You should respect the following constraints

- Deadline for code : **Friday 21 February 2014 1pm**.
- Deadline for report : **Friday 21 February 2014 2pm**.
- This assignment is **mandatory**.
- This assignment must be completed **in groups of two**.

### Modalities

- For each assignment you will need to **return** (before the deadlines) : your **Comet code** and a **report** (pdf and hardcopy) .
- The files for the different assignments can be downloaded from the iCampus site.
- A **hardcopy of your report** must be turned in (box in front of the INGI secretary) before the report deadline.
- Register your group in iCampus.
- Submit your files, **code and report pdf**, on iCampus in a zip (or tar.gz) with the name **tp1_gXY** where **XY** is your group number.
- In case of problems, write to `f.aubry@uclouvain.be`.

# 2 Problems

## 2.1 Nightmare in INGI

Someone in INGI killed Alice. We want to find out who the killer is. Here is what we know :

(a) Exactly three people were in INGI at the time of the murder : Alice, Bob and Chuck.

(b) The killer hates his victim.

(c) Nobody is taller than himself.

(d) If A is taller than B, then B is not taller than A.

(e) The killer is taller than his victim.

(f) Bob hates no one that Alice hates.

(g) Alice hates everybody except Chuck.

(h) Chuck hates everyone that Alice hates.

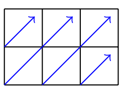
(i) Nobody hates everyone.

We provide you with a commented file `Mystery.co` that models facts (a) to (d).

1. Add the remaining facts to the model
2. Why does the following line result in an execution error ?
   ```
   forall(a in Persons : hates[alice,a] == 1)
     cp.post(hates[chuck,a] == 0)
   ```
3. Write the lines you had to add in your report along with who killed Alice
4. Provide the number of solutions returned by your code. Are there different possible killers ?

## 2.2 Discrete Tomography

Storing an $n$ by $m$ image takes $n \cdot m$ pixels. If the image is black and white an idea to reduce the amout of storage is to store the number of black pixels in each row and the number of black pixels in each colum. This provides a way to store the image using $n + m$ integers. However when reconstructing the image there can be several solutions. To try to reduce the number of solutions we decided to also compute the sum of each diagonal. We only consider the diagonals with positive slope. For instance, with $n = 2$ and $m = 3$ we consider 4 diagonals :



We provide you with a commented file `Tomography.co`. You have to fill in the constraints in order to recover all possible images. We provide 4 instances to test your solution.

Your report must contain :
– The constraints you used.
– The solutions to the 4 instances.

## 2.3 NQueens

Consider a classical NQueens problem. It consists in placing $n$ queens on an $n \times n$ chess-board, such that no queen attacks any other queen. Obviously each queen will have to be placed in a row and a column of her own, these facts are implicitly accounted for in the classical NQueens model. It uses $n$ variables $x_1, \ldots, x_n$, each with the domain $[1 \ldots n]$, where $x_i$ denotes the row position of the queen placed in the $i$th column of the chess board. This problem is specified in the file `NQueens.co`.

Modify this file in order to try different search strategies :

1. Modify the code of the search procedure such as to assign only values that are still in the domain of the variables.
2. Add a `by` statement to the `forall` loop in order to first assign the variables with the smallest number of values in their domains
3. Give the search procedure to first assign the queens in the odd columns.
4. Give the search procedure to first assign the queens in the odd columns and first try to assign values that are less present in the domain of the other variables.
5. Give the search procedure to split (in 2) the domain of an unbound queen.
6. Give the search procedure to assign the queens from the sides to the center (so first the queen 1, then queen $N$, then queen 2,...)

Indicate for each of these points **only** the (**complete**) search procedure in the report.

## 2.4 Bin Packing

This question requires material covered in Lesson 3 on Modelling.

In the Bin Packing problem we have $m$ bins of capacity $C$ and $n$ objects each with a weight $w_i$. We want to put all objects in the bins without exceeding the capacity of any bin. We provide you an elementry model of this problem in the file `BinPacking.co`. That model encodes an instance with 7 objects of weights

$$6, 5, 5, 4, 4, 4, 2$$

and 3 bins each of capacity 10.

1. Explain intuitively what is a domain consistency for a CSP.
2. Suppose that we put objects number 2 and 4 into the first bin. What are the domains of the variables after applying domain consistency on this CSP ? *(Hint : use Comet to compute this by labeling the bins of these two objects.)*
3. Explain what a redundant constraint is. What are the advantages and disadvantages of such constraints ?
4. Add the following constraint to the model :

$$\sum_{b=1}^{m} load[b] = \sum_{i=1}^{n} weight[i]$$

   Perform the same experiment as in 2. and comment on the results.