Constraint Technology (course 1DL023)
Autumn 2007 – Assignment 1

Pierre Flener

Department of Information Technology, Uppsala University, Sweden

— Hard deadline: 08:00 am on Monday 17 September 2007 —

Submission Instructions

1. This assignment is to be solved in teams of at most two students. Take our warning on plagiarism very seriously. *We assume that by submitting a solution you are certifying that it is solely the work of your team, and that everyone on the team contributed to elaborating this solution.*

2. Write clear answers. Justify all answers, except where explicitly not required. State any assumptions you make.

3. Document each program by mathematically describing its model and any relevant features, such as its branching heuristics, search technique, and any special data structures, as well as by giving instructions on how to compile and run it. Include sample test runs with inputs and outputs and check whether these test runs are reproducible by the programs you submit. Verify whether you are using version 1.0.1 of Gecode/J.

4. Put your answer files into a folder named *Surname.Firstname–A*1 or $Surname_1.Firstname_1$–$Surname_2.Firstname_2$–*A*1. Do not use any special characters in the folder name. Include only source code files, that is remove any executable files. Textual answers must be in *.pdf* or *.txt* format. Include a *ReadMe.txt* file to explain the purpose of each other file. Comply strictly with any answer filenames imposed by the questions. Make a *.tgz* or *.zip* compressed archive of maximum 1MB from this folder. *Verify whether it decompresses properly, reproducing your folder exactly, and actually corresponds to Assignment 1.*

5. Submit your compressed folder via the Course Manager server (whose clock may differ from yours) by the deadline given above. This deadline is hard: *No* exceptions will be made. *No* other method of submission will be accepted. Late solutions will be penalised by 2 points for each 24h of delay, but solutions that are late by more than 48h will get 0 (zero) points.

6. When working in a team, *each* team member must submit a solution. Only one of the submitted solutions of a team will be graded, so make sure they are identical. The lateness penalty, if any, for a team will be determined by the moment of its *last* submitted solution.

Failure to follow the instructions above may result in 0 (zero) points, as we reserve the right to process your solutions mechanically.

Question 1: n-Queens Revisited (2 points)

We have seen the probably most popular model of the n-queens problem (see `http://mathworld.wolfram.com/QueensProblem.html`), based on three *distinct* constraints, without any implied constraints or symmetry-breaking constraints. This model is implemented in the `Queens.java` program at `http://www.it.uu.se/research/group/astra/gecode/examples/` (also available in the Gecode/J distribution).

First copy the entire `examples` folder, compile `Queens.java` via `javac examples/Queens.java`, and run it via `java examples/Queens` n. For how large n can it find a first solution within five CPU minutes on your computer? (Hint: `java examples/Queens` n `-nogui -mode time` gives the CPU runtime in milli-seconds without displaying any solutions, and `java examples/Queens -help` displays other command-line options.)

Now consider the 0/1-model where each position on the chess board is represented by a decision variable $q_{ij} \in \{0, 1\}$ such that $q_{ij} = 1$ if and only if there is a queen on row i and column j. State the needed constraints in mathematical notation. Implement this model in Gecode/J as a new program called `queens01.java`. Experiment with several predefined branching heuristics and identify the seemingly best one. You may find the functions in the file `http://www.it.uu.se/research/group/astra/gecode/help-A1.txt` useful.

Compare the CPU runtimes of the two programs in a table for several representative values of n up to the value of n where the faster program can find a first solution within five CPU minutes on your computer. Which program is more time efficient? Why do you believe this is the case?

Question 2: Magic Hexagon ($3 \cdot 1 = 3$ points)

We have seen the magic square problem. This question considers a related problem, where the layout is hexagonal (see `http://mathworld.wolfram.com/MagicHexagon.html`):

```
    A B C
  D E F G
H I J K L
  M N O P
    Q R S
```

with 19 letters. The task is to find an assignment to each letter of an integer within $\{1, \ldots, 19\}$ such that any two letters take distinct values and the sums of all rows and diagonals are the same: $\mathtt{A} + \mathtt{B} + \mathtt{C} = \mathtt{D} + \mathtt{E} + \mathtt{F} + \mathtt{G} = \cdots = \mathtt{A} + \mathtt{D} + \mathtt{H} = \mathtt{B} + \mathtt{E} + \mathtt{I} + \mathtt{M} = \cdots = \mathtt{H} + \mathtt{M} + \mathtt{Q}$.

a. Write a Gecode/J program called `hexagon1.java` using just the needed summation and *distinct* constraints. For the variable ordering, apply the first-fail principle. What predefined value ordering performs best? Why do you believe this is the case? Give the CPU runtime and number of failures (hint: use just the `-nogui` option) your program needs to find a first solution on your computer, and state that solution. How many solutions are there?

b. Derive implied constraints in a way similar to the one for the magic square problem. Add these constraints to `hexagon1.java`, giving a new program called `hexagon2.java`. Give the CPU runtime and number of failures it needs to finds a first solution on your computer, and state that solution. Is the new program more efficient? Why do you believe this is the case? Is the first solution still the same? Why do you believe this is the case?

c. Identify the symmetries of the problem and state suitable symmetry-breaking constraints in mathematical notation. Add these constraints to `hexagon1.java` and `hexagon2.java`, giving two new programs called `hexagon13.java` and `hexagon23.java`, respectively. Give the CPU runtimes and numbers of failures they need to find a first solution on your computer, and state those solutions. Are the new programs more efficient? Why do you believe this is the case? Are the first solutions still the same? Why do you believe this is the case? How many solutions are there now?

Question 3: Consistency ($2 \cdot 1 = 2$ points)

Consider the following constraint satisfaction problems (CSPs):

$$P_1 = \langle X < Y,\ X + Y < 8,\ Y > Z;\ X \in \{2, \ldots, 10\},\ Y \in \{0, \ldots, 10\},\ Z \in \{3, \ldots, 10\} \rangle$$

$$P_2 = \langle X + Y < Z,\ X < Y,\ \mathit{distinct}(X, Y, Z);\ X, Y, Z \in \{1, \ldots, 5\} \rangle$$

Answer the following questions, showing step by step how the constraint stores evolve:

a. Make P_1 arc consistent. Make P_1 bounds consistent. Are the results different? Why?

b. Make P_2 *hyper*-arc consistent. Make P_2 bounds consistent. Are the results different? Why?

Handle the constraints and decision variables in the sequence in which they appear in the CSPs.

Question 4: Stores and Propagators ($3 \cdot 1 = 3$ points)

a. Consider the following constraint stores:

$$s_1 = \{x \mapsto \{1, 2\}, y \mapsto \{2, 3\}\}$$

$$s_2 = \{x \mapsto \{1\}, y \mapsto \{2, 3\}\}$$

$$s_3 = \{x \mapsto \{4\}, y \mapsto \{2, 3\}\}$$

$$s_4 = \{x \mapsto \{1, 2, 4\}, y \mapsto \{2, 3, 5\}\}$$

For all $1 \leq i \neq j \leq 4$, decide whether $s_i < s_j$. List all assignments a such that $a \in s_1$.

b. Given $V = \{x, y\}$ and $U = \{0, \ldots, 5\}$, consider the function $p_=$ defined on stores $s : V \to 2^U$ as follows:

$$p_=(s) = \left\{ \begin{array}{l} x \mapsto s(x) \cap s(y) \\ y \mapsto s(x) \cap s(y) \end{array} \right\}$$

Prove that $p_=$ is a propagator (that is, prove that it is contracting and monotonic). Prove that $p_=$ implements the $x = y$ constraint.

c. Given $V = \{x, y\}$ and $U = \{0, \ldots, 10\}$, consider the function p defined on stores $s : V \to 2^U$ as follows:

$$p(s) = \left\{ \begin{array}{l} x \mapsto \{a \mid a \in s(x) \wedge \exists b \in s(y) : b = 3 - a\} \\ y \mapsto \{b \mid b \in s(y) \wedge \exists a \in s(x) : a = 3 - b\} \end{array} \right\}$$

Prove that p is a propagator. What constraint does p implement? Why?