Constraint Programming: Assignment no. 5

Marco Kuhlmann, M.Sc., Dipl.-Inform. Guido Tack

`http://www.ps.uni-sb.de/courses/cp-ss07/`

This is the second of four graded assignments. The submission deadline is Monday, May 21st, 23:59 CEST. Please submit solutions for the individual exercises as source code, packed into a `tar.gz` or `zip` archive, to `tack@ps.uni-sb.de`.

In this week's assignment, you will implement propagators in Gecode/J.

Exercise 5.1 (Simple propagators, 6P)

In the lecture, you have seen a Gecode/J propagator for $x \neq y$. Implement propagators for $x < y$ and $\max(x, y) = z$. Use the skeletons we provide and just fill in the missing parts (markes by `TODO` comments).

Exercise 5.2 (Linear equations & generic propagators, 10P)

You have seen the basic propagation rules for linear equations in the lecture. We now want to implement linear equations in a slightly more advanced way, using *generic propagators*.

A generic propagator does not operate on variables, but on *variable views*. You have already seen this in the context of Gecode/J, where propagators are defined in terms of `IntVarViews`. A view can, as for `IntVarView`, just provide a read/write interface to a variable. It can, however, also slightly change how the variable behaves.

A *scale view*, e.g., has the same interface as an `IntVarView`. However, it makes a variable x look like a scaled version, ax, for some integer constant a.

A generic propagator now is parametric with respect to the view type it supports.

a) Implement the missing operations in `ScaleView.java` as indicated by the `TODO` comments.
b) Implement a propagator for $\sum_{i=1}^{n} x_i = c$ by completing the skeleton provided in `LinearSum.java`. If you now instantiate this propagator with ScaleViews instead of `IntVarViews`, this gives you a propagator for $\sum_{i=1}^{n} x_i = c$!
c) Test your implementation using the *send more money* example.

Exercise 5.3 (Magic sequence, 14P)

In this exercise you will implement several models for finding magic sequences. Apart from the basic model, you will add implied constraints and also implement your own propagator.

A magic sequence of length n is a sequence of numbers $\langle x_0, x_1, \ldots, x_{n-1} \rangle$ such that when the variable x_i has the value n_j, then the sequence contains exactly n_j i's. For example, the sequence $\langle 1, 2, 1, 0 \rangle$ is a magic sequence, since there is one 0 (and x_0 has the value 1), there are two occurrences of 1, one occurrence of 2, and no occurrences of 3.

a) Your first task is to implement a basic model for finding magic sequences. The script should take an argument specifying the length of the sequence to find. Use one VarArray containing n variables with an appropriate initial domain. To implement the constraint connecting variables to occurrences, you should use reification. This can be done for each value v by defining boolean variables b_i, where the variable b_i is true if and only if the variable x_i has the value v. Then the sum of the boolean variables will be the number of occurrences of the value v.

b) The next task is to improve the script by two implied constraints.
   - A simple argument will show you what the sum of all the variables in the sequence should be. Use this fact to add an implied constraint to the model.
   - Another interesting implied constraint is defined by the following equation.

$$\sum_{i=0}^{n-1} (i-1) \cdot x_i = 0$$

     This equation holds for every magic sequence (why? You don't have to submit an answer, but think about it).

c) Your third task is to implement a specialized propagator for the constraint that you used reification for in the first task. The constraint is called exactly, and posting the constraint for a VarArray `xs`, a value `v`, and an IntVar `y` means that exactly `y` of the variables in `xs` should take the value `v`. Note that you should *not* use the (more general) count-propagator that is already present in Gecode/J.
   You can use `NaryOnePropagator` as the base class for this propagator.
   You should have the same search-space (the same tree in Gist) if you replace the reified constraints with your own propagator.