Exercise 2

Foundations of Constraint Programming

Prof. Michael Thielscher, Sebastian Voigt

International Master Program in Computational Logic — winter term 2008/2009

Date of exercise: 05.11.2008

Exercise 2.1

Consider the following Boolean constraints (see also Slide 22 in Chapter 2):

$$i_1 \wedge o_2 = y_1$$
$$i_2 \wedge o_1 = y_2$$
$$\neg y_1 = o_1$$
$$\neg y_2 = o_2$$

For the above constraints show two successful derivations using the Boolean constraint propagation rules given on Slide 23. For each derivation step you should underline the selected constraint and give the used rule. The initial CSPs are:

$$\langle i_1 \wedge o_2 = y_1, i_2 \wedge o_1 = y_2, \neg y_1 = o_1, \neg y_2 = o_2; i_1 = 0, i_2 = 1 \rangle$$

$$\langle i_1 \wedge o_2 = y_1, i_2 \wedge o_1 = y_2, \neg y_1 = o_1, \neg y_2 = o_2; o_2 = 1, i_1 = 1 \rangle$$

If you see i_1, i_2 as input lines and o_1, o_2 as outputs, can you imagine what kind of electric circuit the above constraints represent?

Exercise 2.2

Consider the CSP from Slide 33 in Chapter 2:

$$\langle x \cdot y = z; x \in [1..20], y \in [9..11], z \in [155..161] \rangle$$

Transform this CSP using the three Multiplication Rules from Slide 32 until no further rule application is possible. Give the selected constraint and the used rule for each derivation step.

Exercise 2.3

Download and install the open source Prolog version Eclipse-Prolog. It can be found together with detailed documentation at the url `http://www.eclipse-clp.org/`. After installation you can start Eclipse-Prolog using the command `eclipse` and load a file *filename.pl* via `compile(filename).`.

Write a program in Eclipse-Prolog that solves the N Queens problem (see Slide 12 in Chapter 1). Use the constraint solving library `fd` for finite domains (load it with `:- lib(fd).`). Constraints are given using the `#` operator, so for example, `N1 #<= N2` would imply that number `N1` is smaller or equal to number `N2`. Use the built-in constraint propagation so that your program is able to find all solutions of an 8-queens problem almost instantaneously.
Compare the run time of your program to the time needed to find one solution of the 8-queens problem without the use of constraint propagation.