Exercise 5

Foundations of Constraint Programming

Prof. Steffen Hölldobler, Sebastian Haufe

International Master Program in Computational Logic — winter term 2010/2011

Date of exercise: 07.01.2011

Exercise 5.1

Reconsider the two CSPs with variables x, y, z, w from Exercise 4.2:

1) $\langle x \neq y, z = x \cdot y \ ; \ x \in \{a, b\}, y \in \{b, c\}, z \in \{bb, cc\} \rangle$, where $\cdot$ is the string concatenation

2) $\langle x \neq 10, x = y + 1, \mathit{all\_different}(x, y, z), x + y + z = w;$
$x \in [10 \ldots 13], y \in [10 \ldots 12], z \in [10 \ldots 12], w \in [30 \ldots 32] \rangle$

a) Can you find some k for which these CSPs are not k-consistent?

b) A CSP is called *regular* if for each sequence X of its variables a unique constraint on X exists. Give a regular ASP for each of the CSPs 1) and 2).

c) For each of the CSPs in 1) and 2) give three different tuples (i, m) such that the CSP is not relationally (i, m)-consistent.

Exercise 5.2

Consider the proof rules for XOR on Slide 11 of Chapter 5 and a 4-Bit BCD-Code to Gray-Code Converter. (Hint: BCD-Code means Binary Coded Decimal and is the usual way to encode a decimal; whereas the n-bit Gray Code is an ordering of 2^n binary numbers such that only one bit changes from one entry to the next). The converter has 4 inputs and 4 outputs and is defined with the following Boolean constraints:

$$\begin{aligned} y_1 &= x_1 \oplus x_2, \\ y_2 &= x_2 \oplus x_3, \\ y_3 &= x_3 \oplus x_4, \\ y_4 &= x_4 \end{aligned}$$

For the above constraints compute two successful derivations, just using propagation (no search). You may define and use additional proof rules for XOR. For each derivation step you should underline the selected constraint and give the used rule. The initial CSPs are:

$$< y_1 = x_1 \oplus x_2, y_2 = x_2 \oplus x_3, y_3 = x_3 \oplus x_4, y_4 = x_4; x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1 >$$
$$< y_1 = x_1 \oplus x_2, y_2 = x_2 \oplus x_3, y_3 = x_3 \oplus x_4, y_4 = x_4; x_2 = 1, y_1 = 1, y_3 = 1, y_4 = 1 >$$

Exercise 5.3

The task of this exercise is to implement in Eclipse-Prolog a predicate `myxor/3` which uses constraint propagation and the constraint handling library `ic`. The predicate should allow to resolve constraint satisfation problems like those from the previous exercise.

a) Define a predicate `myxor/3` that implements the two proof rules for `XOR` given on Slide V/11 in the lecture, i.e. `?-myxor(0,0,Z).` and `?-myxor(1,1,Z).` should succeed with `Z=0`, whereas queries like `?-myxor(X,0,Z).` or `?-myxor(X,Y,1).` should produce a delayed goal. Further queries for testing:

   `?-myxor(X,0,Z), X#::0.` should give answer `Z=0`.
   `?-myxor(0,Y,Z), Y#::1.` should proceed with the delayed goal `myxor(0, 1, Z{[0, 1]})`.

   Hint: Use the predefined predicate `ground/1` which succeeds iff the given argument is ground. Moreover you need the predefined predicate `suspend(Goal,Prio,CondList)` which delays the Goal and wakes it with priority Prio as soon as one of the specifications in CondList occurs. For example `suspend(p(X,Y),2,[X,Y] -> inst)` delays the goal `p(X,Y)` until one of the variables `X,Y` gets instantiated (which is formulated via `[X,Y] -> inst`).

b) Extend the predicate `myxor/3` such that it is able to handle the second CSP from Exercise 5.2, i.e. implement the rules which had to be introduced in order to solve it. Test the predicate by formulating queries that represent the two example CSPs from Exercise 5.2. Moreover:

   `?-myxor(X,0,Z).` should give answer `Z=X`.
   `?-myxor(0,Y,Z), Y#::1.` should give answer `Z=1`.