Exercise 3

Foundations of Constraint Programming

Prof. Michael Thielscher, Sebastian Voigt

International Master Program in Computational Logic — winter term 2009/2010

Date of exercise: 02.12.2009

Exercise 3.1

Consider the following two CSPs with variables x, y, z, w :

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

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

Are these CSPs consistent, node consistent, arc consistent, directionally arc consistent, hyper-arc consistent, path consistent, directionally path consistent? Can you find some k for which these CSPs are not k-consistent ? Explain your answers.

Exercise 3.2

Prove the Note on Slide IV/15: A normalized CSP is path consistent iff for each subsequence x, y, z of its variables

$$C_{x,y} \subseteq C_{x,z} \cdot C_{y,z}^T, \qquad C_{x,z} \subseteq C_{x,y} \cdot C_{y,z}, \qquad C_{y,z} \subseteq C_{x,y}^T \cdot C_{x,z}.$$

Exercise 3.3

a) Implement an Eclipse-Prolog-predicate `permutation(A,B)` that generates a permutation `B` of a list `A`. All permutations can be computed by backtracking using ";" when prompted. Use the following queries to check that your program works correctly:

```
:-permutation([1,2,3],B).
:-permutation(A,B).
```

b) Use the constraint solving library `ic` to write a predicate `sorted(L)` for a list `L` that is using constraints and is true if the elements of the list are sorted in ascending order.

Implement a predicate `permsort(L,SL)` that takes a list `L` and generates a sorted list `SL` by testing permutations until a sorted permutation is found.

c) Redefine `permsort` such that it first calls `sorted` and then `permutation`. Compare the two ways of implementing `permsort` with respect to run time.

Check that `permsort` also works for lists with variables, for example `permsort([1,A,3],SL)`.