Constraint Programmierung
Prof. Dr. Thom Frühwirth, Marc Meister

WS4 – Blatt 4
Besprechung am 23. Nov., 14 Uhr, O27/2201
Veranstaltungsseite `http://www.informatik.uni-ulm.de/pm/index.php?id=82`

Dr. Christian Holzbaur (Wien) spricht zum Thema
A generalized runtime system for Constraint Handling Rules
(morgen [!]) am Mittwoch, den 17. November, 15 Uhr s.t., O27/121.

Aufgabe 11 (Warmup – generate and test)**.**

a) Implement a Prolog-predicate `permutation/2` that generates permutations: `permutation(A,B)` generates a permutation `B` of a list `A` with fixed length. All permutations can be computed by backtracking using ";" when promted. Do not use built-in predicates for manipulating lists. Describe your observations with the following queries:
   - `permutation([1,2,3],[L,M,N]).`
   - `permutation([L,M,N],[1,2,3]).`
   - `permutation([1,2,3],B).`
   - `permutation(A,[1,2,3]).`

b) Improve your program, such that *all four* queries from a) work.

Aufgabe 12 (n-Queens Example – generate and test)**.**
Place n queens on an $n \times n$ chessboard, s.t. no queen attacks any other queen.
As each row must contain exactly one queen, a representation can be given by a list (length n) of X-coordinates. Generate all possible positions and check. If the placement is not save, backtrack.
Write a a Prolog-program `solve(`n`,L)` which solves the n-queens problem for arbitrary n and returns a list `L` of X-coordinates of the n queens.
Hint: The solution given in the introduction uses CHR code – here we try it in Prolog.

Aufgabe 13 (CLP – generate and test vs. constrain and generate)**.**
Sortieren Sie eine Liste von Zahlen mit dem Algorithmus *permutation sort*. Analysieren Sie jeweils das Laufzeitverhalten der Implementierung in Abhängigkeit von der Listenlänge.

a) Implementieren Sie dazu als Prolog-Prädikate `permsort(List,Sorted)`, `permutation(List,Sorted)` und `sorted(Sorted)` folgenden Sachverhalt über Listen:
   Eine Liste R ist die sortierte Version der Liste L wenn R eine Permutation von R ist und die Elemente von R aufsteigend sortiert sind.

b) Laden Sie als Constraintlöser das Modul `clpq` mit `:- use_module(library(clpq)).` und lesen Sie dazu im SICStus-Handbuch – Kapitel 33 – nach.
   Zum Beispiel vereinfacht der Constraintlöser die (in geschweiften Klammern zu schreibende) Constraints `{A=<3, A=<5}` zu `{A=<3}`.
   Schreiben Sie die expliziten und auch die impliziten Vergleichsoperationen aus a) explizit in geschweifte Klammern und rufen Sie in `permsort` zuerst `sorted` und dann `permutation` auf.
   Testen Sie auch Eingabelisten mit Variablen (z.B. `permsort([1,A,3],Sorted)`) und gesetzter Ausgabeliste (z.B. `permsort(X,[1,3,7])`)?
   Wie können Sie eine eventuelle Endlosrekursion verhindern?