CHR programming contest 2010
Warm-up problems

Jon Sneyers and Peter Van Weert

31 August 2010, Leuven

1 Factorial

As input you get `f(1),f(2),...,f(`n`)`. As output we want one constraint `f(`$n!$`)`. Remember that $n!$ is defined as follows: $n! = n(n-1)(n-2)\ldots 1$.

2 Equality constraint solver

Remember the less-than-or-equal (`leq`/2) solver. This is a solver for partial order relations. Partial orders are reflexive, antisymmetric and transitive.

```
reflexive     @ leq(A,A) <=> true.
idempotent    @ leq(A,B) \ leq(A,B) <=> true.
antisymmetry @ leq(A,B), leq(B,A) <=> A=B.
transitivity @ leq(A,B), leq(B,C) ==> leq(A,C).
```

Modify the above program to make a solver for equality (`eq`/2). Equality is an equivalence relation, so it is reflexive, symmetric and transitive.

3 Counting items

You are given as input a sequence of items in `i`/1 constraints, for example `i(bread), i(cheese), i(water), i(bread)`. Your task is to count the items. The output is as `n`/2 constraints, for example `n(bread,2), n(cheese,1), n(water,1)`.

4 Shortest paths

The following program takes a directed graph, where the edges are represented as `e`/2 constraints (`e(A,B)` means that there is a (directed) edge from `A` to `B`), and computes the reachability relation `p`/2 (where `p(A,B)` means that there is some path from `A` to `B`).

```
e(X,Y) ==> p(X,Y).
p(X,Y) \ p(X,Y) <=> true.
e(X,Y), p(Y,Z) ==> p(X,Z).
```

Modify the above program such that it computes the distance relation `d`/3, where `d(A,B,N)` means that the shortest path to go from `A` to `B` uses `N` edges.

Hints for the actual contest

We suggest the following order to solve the actual contest: first problem 4, then problem 6, then problem 1, then problem 2, then problem 5 and then problem 3.

Here are some hints for each problem:

Problem 1: Start from the following Sudoku solver program, (it takes input as `sudoku/1` in the same way as requested):

`http://dtai.cs.kuleuven.be/CHR/summerschool/contest/sudoku.chr`

Problem 2: Here is a high-level algorithm sketch:

For the input `board(X,Y,_)`, it suffices to compute all prime numbers below `X+Y`. You can use the following program to generate prime numbers:

```
primes(1) <=> true.
primes(N) <=> N1 is N-1, prime(N), primes(N1).
prime(A) \ prime(B) <=> 0 is B mod A | true.
```

Create all board positions using a nested loop (in Prolog or CHR) — make board positions as `pos(X,Y,P)` constraints, where `(X,Y)` is the position, and `P` is a variable that gets assigned `'K'` or `'.'`. Try all possible assignments using Prolog disjunction. Fail if you find a conflict (some knight that can take some other knight). Print out the board layout using another nested loop. You can use a failure-driven loop to enumerate all possible board layouts: just add "`,fail`" after the call to the printing routine so it backtracks until it has found all solutions.

Problem 3: You can compute the distance `D` between `(Ax,Ay)` and `(Bx,By)` as follows:

```
distance(Ax,Ay,Bx,By,D) :- D is sqrt((Ax-Bx)^2 + (Ay-By)^2).
```

Use iterative deepening to find the optimal solution (first try to find a solution with 1 bomb, if none is found, try 2 bombs, 3 bombs, and so on).

Problem 4: Should be easy.

Problem 5: Note that the subset relation is a partial order.

Problem 6: Note that preferences are a strict partial order for "rational" supporters (transitive, irreflexive, antisymmetric) but "irrational" supporters have reflexive or symmetric preferences.