Exercices in Coq

Yves Bertot

August 23, 2005

1. Define a function of type `forall x:nat, {y:nat|2*y<=x<2*y+1}`,
2. Define a function `twopower` of type `nat -> nat` that computes 2^n for every natural number n, and then define a function of type

   ```
   forall x:nat,
         {y : nat | twopower y <= x < twopower(y+1)}+{x=0}
   ```

3. Define a sorting function for an abitrary binary relation on an arbitrary type. You should define suitable predicates `sorted` and `permutation`, and then provide a function with the following type:

   ```
   forall A:Set, forall R:Set,
   forall testR : forall x y,{R x y}+{R y x},
   forall l : list A,
     {l' : list A | sorted A R l' /\ permutation A l l'}
   ```

   I suggest using insertion sort, which is reasonably simple. As a first exercise, you may leave aside the notion of permutation and construct a function that only has the following type:

   ```
   forall A:Set, forall R:Set,
   forall testR : forall x y,{R x y}+{R y x},
   forall l : list A, {l' : list A | sorted A R l'}
   ```

   But because this specification is weak, you should refrain from cheating (for instance by providing the constant function that returns the empty list).