Coq Exercises

MAPi – PSVC

Porto, 2008

Please solve the following exercises *without* using automatic tactics such as auto, tauto, omega, etc.

1 Logical Reasoning

1. Prove the following propositional formulas (P, Q : Prop):
   (a) $(P \to Q) \to \sim Q \to \sim P$
   (b) $(P \lor Q \to R) \to (P \to R)$
   (c) $(P \lor Q) \land \sim P \to Q$
   (d) $P \land Q \to R \leftrightarrow P \to Q \to R$
   (e) $\sim(P \lor Q) \leftrightarrow \sim P \land \sim Q$
   (f) $\sim\sim(P \lor \sim P)$
2. Prove the following theorem:

   Theorem *ex4* : $\forall$ (X : Set) ($P : X \to$ Prop), $\sim(\forall\ x, \sim(P\ x)) \to$ (*exists* x, $P\ x$).

3. Assuming the *Excluded Middle* axiom, prove:
   (a) **Theorem** *Pierce* : $\forall\ P\ Q, ((P \to Q) \to P) \to P$.
   (b) **Theorem** *NNE* : $\forall\ P, \sim\sim P \to P$.

2 Natural Numbers

Recall the definition of natural numbers as an inductive type (command Print *nat*).

1. Define a function *add* : *nat* $\to$ *nat* $\to$ *nat* that computes the sum of two natural numbers.
2. Prove commutativity of *add* (Hint: use *lemmas* to isolate interesting proof obligations).
3. Prove the following fact: *for a natural number n, $n * n$ can be computed as the sum of the first n odd numbers (e.g. $3 * 3 = 1 + 3 + 5$).* (Hint: define first a function that sums the odd numbers. You should also use definitions/facts/theorems from the library *Arith* — try also the omega tactic).

3 Reasoning about lists

The following exercises require the library `Lists`. You can load that library by executing the command Require Import *Lists*.

1. Consider the following inductive relation:

   Inductive $last\ (A : \mathsf{Set})\ (x : A) : list\ A \to \mathsf{Prop} :=$
   $\mid last\_base : last\ x\ (cons\ x\ nil)$
   $\mid last\_step : \forall\ l\ y, last\ x\ l \to last\ x\ (cons\ y\ l).$

   (a) Use `inversion` to prove that $\forall\ x, \sim(last\ x\ nil)$.

   (b) (Difficult) Try to avoid using that tactic.

2. Consider the following definition for the `Even` predicate:

   Inductive $Even : nat \to \mathsf{Prop} :=$
   $\mid Even\_base : Even\ 0$
   $\mid Even\_step : \forall\ n, Even\ n \to Even\ (S\ (S\ n)).$

   (a) Define, accordingly, the `Odd` predicate. Prove that, for every number n, $Even\ n \to Odd\ (S\ n)$.

   (b) Define the function `rev` that reverses a list.

   (c) Prove that, for every list l, $rev\ (rev\ l) = l$.

   (d) Recall the definition for the function `app` (concatenation of lists). Prove that for every lists l_1 and l_2, $rev\ (app\ l1\ l2) = app\ (rev\ l2)\ (rev\ l1)$.

3. Consider the inductive predicate:

   Inductive $InL\ (A : \mathsf{Type})\ (a : A) : list\ A \to \mathsf{Prop} :=$
   $\mid InHead : \forall\ (xs : list\ A), InL\ a\ (cons\ a\ xs)$
   $\mid InTail : \forall\ (x : A)\ (xs : list\ A), InL\ a\ xs \to InL\ a\ (cons\ x\ xs).$

   Prove the following properties:

   (a) $\forall\ (A : \mathsf{Type})\ (a : A)\ (l1\ l2 : list\ A), InL\ a\ l1 \lor InL\ a\ l2 \to InL\ a\ (app\ l1\ l2)$.

   (b) $\forall\ (A : \mathsf{Type})\ (a : A)\ (l1\ l2 : list\ A), InL\ a\ (app\ l1\ l2) \to InL\ a\ l1 \lor InL\ a\ l2$.

4. Define the function `elem` that checks if an element belongs a list of integers (Hint: import *ZArith* module in order to use the *Z_eq_dec* that tests for integer equality).

5. Prove the correctness of `elem`, that is, $\forall\ (a : Z)\ (l1\ l2 : list\ Z), elem\ a\ (app\ l1\ l2) = orb\ (elem\ a\ l1)\ (elem\ a\ l2)$ (the function `orb` is the boolean-or function defined in Library `Bool`).