Exercícios de Coq

MFES 2009/10

June 30, 2010

Part I

Apresente provas em Coq dos seguintes resultados. Nesta primeira parte, não use tácticas de prova automática.

Grupo 1

1. $P \wedge (Q \vee R) \rightarrow (P \wedge Q) \vee (P \wedge R)$
2. $(\forall x.P(x) \rightarrow Q(x)) \rightarrow (\exists x.P(x)) \rightarrow \exists y.Q(y)$

Grupo 2

1. $(P \vee Q) \vee R \rightarrow P \vee (Q \vee R)$
2. $(\forall x.Q(x) \rightarrow R(x)) \rightarrow (\exists x.P(x) \wedge Q(x)) \rightarrow \exists x.P(x) \wedge R(x)$

Grupo 4

1. $(P \wedge Q) \wedge R \rightarrow P \wedge (Q \wedge R)$
2. $(\exists x.P(x)) \rightarrow (\forall x.\forall y.P(x) \rightarrow Q(y)) \rightarrow \forall y.Q(y)$

Grupo 5

1. $(Q \rightarrow R) \rightarrow P \vee Q \rightarrow P \vee R$
2. $(\exists x.P(x)) \vee (\exists x.Q(x)) \rightarrow \exists x.P(x) \vee Q(x)$

Grupo 6

1. $P \vee (Q \wedge R) \rightarrow (P \vee Q) \wedge (P \vee R)$
2. $(\exists x.P(x) \vee Q(x)) \rightarrow (\exists x.P(x)) \vee (\exists x.Q(x))$

Grupo 7

1. $(P \wedge Q) \vee (P \wedge R) \rightarrow P \wedge (Q \vee R)$
2. $(\exists x.\forall y.P(x,y)) \rightarrow \forall y.\exists x.P(x,y)$

Grupo 8

1. $(P \vee Q) \wedge (P \vee R) \rightarrow P \vee (Q \wedge R)$
2. $(\exists x.P(x) \wedge Q(x)) \rightarrow (\exists x.P(x)) \wedge (\exists x.Q(x))$

Parte II

Das provas que a seguir se propõem, apresente a resolução de pelo menos uma alínea.
Quantas mais alíneas resolver melhor será a sua avaliação. (Pode usar tácticas automáticas.)

Os exercícios que se seguem requerem as bibliotecas `ZArith` e `Lists`.

1. Considere os seguintes predicados definidos indutivamente:

```
Inductive Odd : nat -> Prop :=
| Odd_base : Odd (S 0)
| Odd_step : forall n, Odd n -> Odd (S (S n)).

Inductive Even : nat -> Prop :=
| Even_base : Even 0
| Even_step : forall n, Even n -> Even (S (S n)).
```

   Prove que

   (a) `forall n1 n2, Odd n1 -> Odd n2 -> Even (n1+n2)`

   (b) `forall n1 n2, Even n1 -> Odd n2 -> Odd (n1+n2)`

2. Defina a função `sum` que calcula o somatório de uma lista de inteiros e prove as seguintes propriedades:

   (a) `forall l1 l2, sum (l1 ++ l2) = sum l1 + sum l2`

   (b) `forall l, sum (rev l) = sum l`

3. `forall (A:Type) (l:list A), length (rev l) = length l`

4. `forall (A B:Type) (f:A->B) (l:list A), length (map f l) = length l`

5. Considere o seguinte predicado definido indutivamente:

```
Inductive InL (A:Type) (y:A) : list A -> Prop :=
| InHead : forall xs:list A, InL y (cons y xs)
| InTail : forall (x:A) (xs:list A), InL y xs -> InL y (cons x xs).
```

   Prove as seguintes propriedades:

   (a) `forall (A:Type) (x:A) (l:list A), InL x l -> InL x (rev l)`

   (b) `forall (A B:Type) (y:B) (f:A->B) (l:list A), InL y (map f l) -> exists x, InL x l /\ y = f x`

   (c) `forall (A:Type) (x:A) (l : list A), InL x l -> exists l1, exists l2, l = l1 ++ (x::l2)`

6. Considere o seguinte predicado definido indutivamente:

```
Inductive SubSeq (A:Type) : list A -> list A -> Prop :=
| SubNil : forall l:list A, SubSeq nil l
| SubCons1 : forall (x:A) (l1 l2:list A), SubSeq l1 l2 -> SubSeq l1 (x::l2)
| SubCons2 : forall (x:A) (l1 l2:list A), SubSeq l1 l2 -> SubSeq (x::l1) (x::l2).
```

   Prove as seguintes propriedades:

   (a) `forall (A:Type) (x:A) (l1 l2:list A), SubSeq l1 l2 -> InL x l1 -> InL x l2`

   (b) `forall (A:Type) (l1 l2 l3 l4:list A), SubSeq l1 l2 -> SubSeq l3 l4 -> SubSeq (l1++l3) (l2++l4)`

   (c) `forall (A:Type) (l1 l2:list A), SubSeq l1 l2 -> SubSeq (rev l1) (rev l2)`

   (d) `forall (A B:Type) (f:A->B) (l1 l2:list A), SubSeq l1 l2 -> SubSeq (map f l1) (map f l2)`