Vérification (HAI603I)

Licence Informatique
Département Informatique
Faculté des Sciences de Montpellier
Université de Montpellier

Contrôle continu du 10 mars 2022

Tous les documents et notes de cours sont autorisés. La documentation de Coq en ligne est également autorisée. Aucun autre document n'est autorisé. L'épreuve dure 1h. Le barème est donné à titre indicatif. Le sujet comporte 2 pages et il y a 3 exercices.

Toutes les réponses sont à réaliser en Coq dans un unique fichier que vous rendrez sous Moodle. Attention à bien sélectionner le rendu correspondant à votre groupe.

Exercice 1 (7 pts)

Démontrer en Coq les formules suivantes :

1. $(\forall x.P(x) \Rightarrow Q(x)) \wedge P(a) \Rightarrow \exists x.Q(x)$
2. $(\exists x.P(x) \wedge Q(x)) \Rightarrow (\exists x.P(x)) \vee (\exists x.Q(x))$

où P et Q sont des symboles de prédicat d'arité 1 et a une constante.

Exercice 2 (7 pts)

On considère la structure algébrique des anneaux commutatifs $(R, +, \times)$ définie en Coq de la façon suivante :

```
Parameter R : Set.
Parameters zero one : R.
Parameter opp : R → R.
Parameters plus mult : R → R → R.

Section Commutative_ring.

Variables a b c : R.

Axiom ass_plus : plus (plus a b) c = plus a (plus b c).
Axiom com_plus : plus a b = plus b a.
Axiom com_mult : mult a b = mult b a.
Axiom ass_mult : mult (mult a b) c = mult a (mult b c).
Axiom dis_left : mult a (plus b c) = plus (mult a b) (mult a c).
Axiom dis_right : mult (plus b c) a = plus (mult b a) (mult c a).
Axiom neu_plus_r : plus a zero = a.
Axiom neu_plus_l : plus zero a = a.
```

```
Axiom neu_mult_r : mult a one = a.
Axiom neu_mult_l : mult one a = a.
Axiom opp_right : plus a (opp a) = zero.
Axiom opp_left : plus (opp a) a = zero.

End Commutative_ring.
```

Dans cette structure, démontrer que :

$\forall a, b \in R. 2 \times (a + b) = a + (b + (a + b))$

où 2 est représenté par $1 + 1$.

Exercice 3 (7 pts)

On considère la relation inductive *is_rev* et la fonction *rev* sur les listes d'entiers naturels définies comme suit en Coq :

```
Require Export List.
Open Scope list_scope.
Import ListNotations.

Inductive is_rev : list nat → list nat → Prop :=
| is_rev_nil : is_rev nil nil
| is_rev_cons : forall (n : nat) (l1 l2 : list nat),
  is_rev l1 l2 → is_rev (n::l1) (l2 ++ [n]).

Fixpoint rev (l : list nat) {struct l} : list nat :=
  match l with
    | nil ⇒ nil
    | e::q ⇒ (rev q) ++ (e::nil)
  end.
```

Démontrer que la fonction *rev* est complète vis-à-vis de la relation *is_rev*, c'est-à-dire :

$\forall l1, l2 \in \mathcal{L}.(is\_rev\ l1\ l2) \Rightarrow (rev\ l1) = l2$

où $\mathcal{L}$ représente l'ensemble des listes d'entiers naturels.

La preuve devra se faire par induction structurelle sur la relation $(is\_rev\ l1\ l2)$.