Une Introduction à la Vérification de Programmes avec Coq (draft)

Master Informatique, Histoire et épistémologie du calcul et de l'informatique, Travaux Pratiques

Reynald Affeldt
National Institute of Advanced Industrial Science and Technology

20 février 2016

Table des matières

1 La soustration des entiers naturels avec types dépendants **1**

2 Vérification du programme factoriel avec la logique de Hoare **2**
2.1 Prouver le lemme de correction suivant avec la logique de Hoare vue en cours 2

1 La soustration des entiers naturels avec types dépendants

Sur le modèle de la fonction prédécesseur vue en cours, on va implémenter une soustration $m-n$ sur les entiers naturels qui n'est définie que lorsque $m \geq n$.

Un exemple de fonction totale implémentant la soustraction des entiers naturels :

```
Fixpoint tminus (n m : nat) : nat :=
  match n with
  | O ⇒ O
  | S n' ⇒ match m with
            | O ⇒ n
            | S m' ⇒ tminus n' m'
            end
  end.

Compute tminus 5 3.
Compute tminus 5 6.
```

Implementer une fonction équivalente avec le type suivant :

```
Fixpoint pminus (n m : nat) : m ≤ n → nat.
Abort.
```

```
Require Import Arith.
```

Tester la fonction produite.

```
Lemma O_le_5 : 3 ≤ 5.
```

```
Proof.
auto.
Qed.

Compute pminus 5 3 O_le_5.
```

Implementer une fonction équivalente avec le type suivant :

```
Fixpoint sminus (n m : nat) : m ≤ n → { k : nat | k + m = n }.
Abort.
```

2 Vérification du programme factoriel avec la logique de Hoare

On va utiliser la logique de Hoare définie en cours pour reproduire la vérification de l'exemple de la fonction factorielle. La fonction `fact` de la librairie `Factorial` de la librairie standard de Coq fera office de spécification.

Prouver le lemme de correction suivant avec la logique de Hoare vue en cours

```
Require Import Omega.
Require Import Factorial.

Lemma facto_fact x X ret : x ≠ ret →
  hoare
    (fun s ⇒ eval (exp_var x) s = X ∧ eval (exp_var ret) s = 1)
    (facto x ret)
    (fun s ⇒ eval (exp_var ret) s = fact X).
Proof.
```