# Sémantique des langages de programmation
# Mastère PLMT
# examen

Yves Bertot

Décembre 2004

## 1 Vérification de preuves sur machine

Nous reprenons la définition d'un langage d'expressions arithmétiques donnée par le script Coq suivant:

```
Require Export List.  Require Export ZArith.

Inductive aexpr : Set :=
  avar : Z -> aexpr
| aint : Z -> aexpr
| aplus : aexpr -> aexpr -> aexpr.

Inductive a_eval : list(Z*Z) -> aexpr -> Z -> Prop :=
  ae_int : forall r n, a_eval r (aint n) n
| ae_var1 : forall r x v, a_eval ((x,v)::r) (avar x) v
| ae_var2 : forall r x y v v' ,
              x <> y -> a_eval r (avar x) v' ->
              a_eval ((y,v)::r) (avar x) v'
| ae_plus : forall r e1 e2 v1 v2,
              a_eval r e1 v1 -> a_eval r e2 v2 ->
              a_eval r (aplus e1 e2) (v1 + v2).
```

Nous ajoutons la définition suivante, qui décrit le remplacement d'une variable par une constante:

```
Inductive subst : Z -> Z -> aexpr -> aexpr -> Prop :=
  su_var1 :
    forall var val : Z,
      subst var val (avar var) (aint val)
| su_var2 :
    forall var val x : Z,
      var <> x -> subst var val (avar x) (avar x)
```

```
| su_int : forall var val n:Z,
      subst var val (aint n) (aint n)
| su_plus :
    forall var val e1 e2 e3 e4,
      subst var val e1 e3 -> subst var val e2 e4 ->
      subst var val (aplus e1 e2) (aplus e3 e4).
```

On suppose que le théorème suivant a déjà été démontré:

```
Theorem eval_det :
 forall r e n1 n2, a_eval r e n1 -> a_eval r e n2 -> n1=n2.
Admitted.
```

On se propose de démontrer le théorème suivant:

```
Theorem subst_correct :
 forall var val e1 e2,
 subst var val e1 e2 ->
 forall r: list(Z*Z),
 a_eval r (avar var) val ->
 forall n:Z, a_eval r e1 n -> a_eval r e2 n.
```

A priori, cette démonstration se fait par récurrence sur la dérivation de `subst var val e1 e2`, en utilisant des inversions sur la dérivation de `a_eval r e1 n`. Donnez la séquence de commandes qui permet d'effectuer cette preuve en `Coq` en donnant à chaque étape suffisamment d'information sur le but en cours.

We re-use the definition of arithmetic expressions and their evaluation. We add a new notion, `subst`, that describes the operation of replacing a variable with a constant. We suppose that the theorem `eval_det` is already proved.

We want to prove the statement `subst_correct` given above. A sensible proof relies on an induction on the derivation for `subst var val e1 e2`, using inversions on the derivation of `a_eval r e1 n`. Write down the command sequence that performs this proof in `Coq`, giving at each step enough information about the current goal.