Théorie de la Programmation, L3IF – TD

mercredi 17 octobre 2018

1 Induction

Definitions

Considerez les definitions inductives suivantes :

Inductive expr :

$\texttt{Const}$: nat → expr

$\texttt{Plus}$: expr → expr → expr

$\texttt{IfZero}$: expr → expr → expr → expr

$$\frac{}{eval\ (\texttt{Const}\ n)\ n} \qquad \frac{eval\ a_1\ n_1 \qquad eval\ a_2\ n_2}{eval\ (\texttt{Plus}\ a_1\ a_2)\ (n_1 + n_2)}$$

$$\frac{eval\ a_1\ 0 \qquad eval\ a_2\ n}{eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n} \qquad \frac{eval\ a_1\ (S\ k) \qquad eval\ a_3\ n}{eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n}$$

$$\frac{}{(\texttt{Plus}\ (\texttt{Const}\ n_1)\ (\texttt{Const}\ n_2)) \to (\texttt{Const}\ (n_1 + n_2))} \qquad \frac{a_1 \to a_1'}{(\texttt{Plus}\ a_1\ a_2) \to (\texttt{Plus}\ a_1'\ a_2)}$$

$$\frac{a_2 \to a_2'}{(\texttt{Plus}\ (\texttt{Const}\ n_1)\ a_2) \to (\texttt{Plus}\ (\texttt{Const}\ n_1)\ a_2')} \qquad \frac{a_1 \to a_1'}{(\texttt{IfZero}\ a_1\ a_2\ a_3) \to (\texttt{IfZero}\ a_1'\ a_2\ a_3)}$$

$$\frac{}{(\texttt{IfZero}\ (\texttt{Const}\ 0)\ a_2\ a_3) \to a_2} \qquad \frac{}{(\texttt{IfZero}\ (\texttt{Const}\ (S\ k))\ a_2\ a_3) \to a_3}$$

$$\frac{}{a \to^* a} \qquad \frac{a_1 \to a_2 \qquad a_2 \to^* a_3}{a_1 \to^* a_3}$$

1.1 Exercice 1

Montrez $\forall a\ b, (a \to b \Rightarrow (\forall n, eval\ a\ n \Leftrightarrow eval\ b\ n))$ par induction sur $\to$ en utilisant l'hypothese d'induction $P\ a_1\ a_2 \equiv (\forall n, eval\ a_1\ n \Leftrightarrow eval\ a_2\ n)$.

Correction. Comme énoncé, on procède par induction sur $\to$. Il y a 6 cas à traiter qui correspondent aux 6 règles de $\to$.

- **Cas Plus Constante.** Il faut prouver $\forall n, eval\ (\texttt{Plus}\ (\texttt{Const}\ n_1)\ (\texttt{Const}\ n_2))\ n \Leftrightarrow eval\ (\texttt{Const}\ (n_1 + n_2))\ n$. Par définition de *eval*, la seule dérivation possible de $eval\ (\texttt{Plus}(\texttt{Const}\ n_1)(\texttt{Const}\ n_2)\ n$ est :

$$\frac{\frac{}{eval\ (\texttt{Const}\ n_1)\ n_1} \qquad \frac{}{eval\ (\texttt{Const}\ n_2)\ n_2}}{eval\ (\texttt{Plus}\ (\texttt{Const}\ n_1)\ (\texttt{Const}\ n_2)\ (n_1 + n_2)}$$

  et la seule dérivation possible pour $\texttt{Const}\ (n_1 + n_2)$ est $eval\ (\texttt{Const}\ (n_1 + n_2))\ (n_1 + n_2)$, ce qui conclut ce cas.

- **Cas Plus Gauche.** Il faut prouver $\forall n, eval\ (\texttt{Plus}\ a_1\ a_2)\ n \Leftrightarrow eval\ (\texttt{Plus}\ a_1'\ a_2)\ n$ avec les hypothèses de récurrence $(a_1 \to a_1')$ et $\forall n, eval\ a_1\ n \Leftrightarrow eval\ a_1'\ n$. Soit $n \in \mathbb{N}$. On prouve la première implication. Supposons $eval\ (\texttt{Plus}\ a_1\ a_2)\ n$. Par définition de *eval*, il existe n_1 et n_2 tel que l'on ait $eval\ a_1\ n_1$, $eval\ a_2\ n_2$ et $n_1 + n_2 = n$. Par l'hypothèse de récurrence, on a $eval\ a_1'\ n_1$, et donc on peut dériver :

$$\frac{eval\ a_1'\ n_1 \qquad eval\ a_1\ n_2}{eval\ (\texttt{Plus}\ a_1'\ a_2)\ (n_1 + n_2)}$$

. L'autre implication $eval\ (\texttt{Plus}\ a_1'\ a_2)\ n \Leftrightarrow eval\ (\texttt{Plus}\ a_1\ a_2)\ n$ se prouve de la même manière.

- **Cas Plus Droite.** Il faut prouver $(\forall n, eval\ (\texttt{Plus}\ (\texttt{Const}\ n_1)\ a_2)\ n \Leftrightarrow eval\ (\texttt{Plus}\ (\texttt{Const}\ n_1)\ a_2')\ n$ avec les hypothèses de récurrence $(a_2 \to a_2')$ et $\forall n, eval\ a_2\ n \Leftrightarrow eval\ a_2'\ n$. Ce cas se prouve de facon similaire au cas précédent du plus à gauche.

- **Cas If Contextuel.** Il faut prouver $\forall n, eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n \Leftrightarrow eval\ (\texttt{IfZero}\ a_1'\ a_2\ a_3)\ n$ avec les hypothèses de récurrence $(a_1 \to a_1')$ et $\forall n, eval\ a_1\ n \Leftrightarrow eval\ a_1'\ n$. Soit $n \in \mathbb{N}$. On prouve d'abord la première implication. Supposons $eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n$. Par définition de *eval*, il y a deux règles possibles et donc deux cas à traiter:

  – Supposons que l'on soit dans le cas :

  $$\frac{eval\ a_1\ 0 \qquad eval\ a_2\ n}{eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n}$$

  Par hypothèse de récurrence, $eval\ a_1\ 0$ nous donne $eval\ a_1'\ 0$, et donc en utilisant la même règle, on obtient $eval\ (\texttt{IfZero}\ a_1'\ a_2\ a_3)\ n$

  – Supposons que $eval\ a_1\ (S\ k)$ et $eval\ a_3\ n$. Par hypothèse de récurrence, on a $eval\ a_1'\ (S\ k)$, et donc par définition de *eval*, on obtient $eval\ (\texttt{IfZero}\ a_1\ a_2\ a_3)\ n$

  Ce qui prouve cette implication. La réciproque se prouve de façon similaire.

- **Cas If True.** Il faut prouver $\forall n, eval\ (\texttt{IfZero}\ (\texttt{Const}\ 0)\ a_2\ a_3)\ n \Leftrightarrow eval\ a_2\ n$. Supposons que $eval\ (\texttt{IfZero}\ (\texttt{Const}\ 0)\ a_2\ a_3)\ n$. Par définition de *eval*, on a $eval\ (\texttt{Const}\ 0)\ n \Leftrightarrow (n = 0)$. Ainsi, la seule façon d'avoir $eval\ (\texttt{IfZero}\ (\texttt{Const}\ 0)\ a_2\ a_3)\ n$ est quand on a $eval\ a_2\ n$ et que l'on applique la règle de *eval* correspondante. Ceci prouve la première implication. La seconde implication, $eval\ a_2\ n \Rightarrow eval\ (\texttt{IfZero}\ (\texttt{Const}\ 0)\ a_2\ a_3)\ n$ est directe, il suffit d'appliquer la bonne règle de *eval*.

- **Cas If False.** Il faut prouver $\forall n, eval\ (\texttt{IfZero}\ (\texttt{Const}\ (S\ k))\ a_2\ a_3)\ n \Leftrightarrow eval\ a_3\ n$. Ce cas est totalement similaire au précédent.

1.2 Exercice 2

Maintenant, prouvez que la relation $\to^*$ est transitive.

Correction. Formellement, on doit prouver que $\forall a, b, c, (a \to^* b) \Rightarrow (b \to^* c) \Rightarrow (a \to^* c)$. On prouve ceci par induction sur $a \to^* b$. La propriété P que l'on souhaite prouver est alors $P\ a_1\ a_2 \equiv (\forall c, (a_2 \to^* c) \Rightarrow (a_1 \to^* c))$

- **Cas Reflexif.** On traite d'abord le cas $a_1 = a_2$. Il faut alors prouver dans ce cas là que $(\forall c, (a_2 \to^* c) \Rightarrow (a_1 \to^* c)$ ce qui est trivial quand $a_1 = a_2$.

- **Cas Transitif.** Il faut prouver $(\forall c, (a_2 \to^* c) \Rightarrow (a_1 \to^* c)$ avec les hypothèses d'induction $a_1 \to a_3$, $a_3 \to^* a_2$ et $(\forall c, (a_2 \to^* c) \Rightarrow (a_3 \to^* c)$. Soit c une expression. Supposons que $a_2 \to^* c$. Par hyothèse d'induction, on a que $a_3 \to^* c$. Or, on sait que $a_1 \to a_3$, donc par définition de $\to^*$, on en déduit que $a_1 \to^* c$, ce qui conclut la preuve.

1.3 Exercice 3 (A la maison)

Prouvez que $\forall a, k, (a \to^* \texttt{Const}\ k) \Leftrightarrow eval\ a\ k$

2 Dériver des triplets de Hoare

2.1 Dérivations dans la logique de Hoare

Dériver les triplets suivants:

1. $\{x \geq 0\} x := x + 1;\ x := x + 1 \{x \geq 2\}$
2. $\{\} i := 0;$ `while` $i < 100$ `do` $(i := i + 1) \{i = 100\}$
3. $\{x = a, y = b\} t := x;\ x := y;\ y := t \{x = b, y = a\}$

Correction.

1. $x \geq 0$
   $\Downarrow$
   $x + 1 \geq 1$
   $x := x + 1;$
   $x \geq 1$
   $\Downarrow$
   $x + 1 \geq 2$
   $x := x + 1$
   $x \geq 2$
2. $\Downarrow$
   $0 \leq 100$
   $i := 0;$
   $i \leq 100$
   `while` $(i < 100)$ `do` Invariant : $i \leq 100$

$i \leq 100; i < 100$ $\Downarrow$ $i + 1 \leq 100$ $i := i + 1$ $i \leq 100$

   $i \leq 100, i \geq 100$
   $\Downarrow$
   $i = 100$
3. $x = a, y = b$
   $t := x;$
   $t = a, y = b$
   $x := y;$
   $t = a, x = b$
   $y := t$
   $y = a, x = b$

2.2 Spécification de programmes

Spécifier le comportement attendu des programmes suivants (pré et post-conditions), puis dériver cette spécification dans la logique de Hoare.

Min-Max

```
if y < x do
  t := x;
  x := y;
  y := t
else
  skip
```

Carré

```
i := 0;
s := 0;
while i < n do
  s := s + 2i + 1;
  i := i + 1
```

Mult

```
r := 0;
k := 0;
while k ≤ n do
  r := r + k;
  k := k + 1
```

Correction

1. Pour Min-Max, on prouve $\vdash \{y = a, x = b\}\ Min - Max\ \{y = max(a,b), x = min(a,b)\}$

   $y = a, x = b$

   `if` $y < x$ `do`

   $y = a, x = b, y < x$

   $\Downarrow$

   $y = a, x = b, a < b$

   $t := x;\ x := y;\ y := t$

   $y = b, x = a, a < b$

   $\Downarrow$

   $y = max(a,b), x = min(a,b)$

   `else`

   $y = a, x = b, y \geq x$

   `skip`

   $y = a, x = b, y \geq x$

   $\Downarrow$

   $y = max(a,b), x = min(a,b)$

2. Pour carré, on prouve que $\vdash \{n \geq 0\}\ Carre\ \{i = n, s = n^2\}$

   $n \geq 0$

   $\Downarrow$

   $n \geq 0, 0 = 0^2$

   $i := 0$

   $n \geq i, 0 = i^2$

   $s := 0$

   $n \geq i, s = i^2$

   `while` $i < n$ `do` Invariant : $n \geq i, s = i^2$

$n \geq i, s = i^2, i < n$
$\Downarrow$
$s + 2i + 1 = (i+1)^2, i + 1 \leq n$
$s := s + 2i + 1$
$s = (i+1)^2, i + 1 \leq n$
$i := i + 1$
$s = i^2, i \leq n$

$n \geq i, s = i^2, i \geq n$

$\Downarrow$

$i = n, s = n^2$

3. Pour Mult, prouvons que $\vdash \{n \geq 0\} Mult \{k = n + 1, r = \frac{n(n+1)}{2}\}$

   $n \geq 0$

   $\Downarrow$

   $n + 1 \geq 0, 0 = \frac{0(0-1)}{2}$

   $r := 0$

   $n + 1 \geq 0, r = \frac{0(0-1)}{2}$

   $k := 0$

   $n + 1 \geq k, r = \frac{k(k-1)}{2}$

   `while` $k \leq n$ `do` Invariant : $n + 1 \geq k, r = \frac{k(k-1)}{2}$

   $n + 1 \geq k, r = \frac{k(k-1)}{2}, k \leq n$
   $\Downarrow$
   $r + k = \frac{(k+1-1)(k+1)}{2}, k + 1 \leq n + 1$
   $r := r + k$
   $r = \frac{(k+1-1)(k+1)}{2}, k + 1 \leq n + 1$
   $k := k + 1$
   $r = \frac{(k-1)k}{2}, k \leq n + 1$

   $n + 1 \geq k, r = \frac{k(k-1)}{2}, k > n$

   $\Downarrow$

   $k = n + 1, r = \frac{(n+1)n}{2}$