Université Paul Sabatier
L2 - S4 - Printemps 2005

Programmation impérative et langage C

feuille de **TD n° 5 :** Calculs de plus faibles préconditions et preuves de programmes

---

Comme on l'a vu au TD précédent, à partir de l'énoncé d'un problème à résoudre, on définit sa spécification en triplet :

/* Prédicat d'entrée (PE) */
appel du programme solution
/* Prédicat de sortie (PS) */

Etablir la preuve de la validité d'un programme solution consiste à montrer que si PE est vrai et que le programme solution s'exécute, alors PS sera vrai.
Les exercices qui suivent comprennent des programmes ADA dont on veut prouver la correction.
Le processus de preuve comporte 2 étapes :
– une preuve de correction partielle : si le programme solution s'arrête alors on obtient bien la solution attendue, c'est-à-dire que, si PE est vrai et que le programme solution s'exécute, alors PS est vrai également.
– une preuve de terminaison (en cas de boucle seulement) : le programme s'arrête.
On utilise la notion de "plus faible précondition" (pfp en français, wp en anglais). Prouver un programme solution consiste à vérifier la validité de l'implication logique suivante :

/* PE */ $\longrightarrow pfp\ [\ programme\_solution, /*PS*/\ ]$

1 Cas d'une affectation

On applique la règle suivante :

$pfp\ [x = e, /*Q*/] = /*(Q_e^x) \wedge (e$ évaluable$)*/$ avec Q_e^x le prédicat Q dans lequel on substitue toutes les occurrences de x par e

2 Cas d'une séquence

On commence par traiter la dernière instruction de la séquence et on revient vers la première instruction.

$$\begin{aligned}\text{pfp}[A_1; A_2; \ldots; A_n, /*Q*/] &= \text{pfp}[A_1; A_2; \ldots; A_{n-1}, \text{pfp}[A_n, /*Q*/]] \\ &= \text{pfp}[A_1; A_2; \ldots; A_{n-2}, \text{pfp}[A_{n-1}, \text{pfp}[A_n, /*Q*/]]] \\ &= \ldots\end{aligned}$$

Exemple : soit la séquence d'affectations $x = e_1; y = e_2;$ on a :

$$\begin{aligned}\text{pfp}[x = e_1; y = e_2, /*Q*/] &= \text{pfp}[x = e_1, \text{pfp}(y = e_2, /*Q*/)] \\ &= \text{pfp}[x = e_1, /*(Q^y_{e_2}) \wedge (e_2 \text{ évaluable})*/)] \\ &= /*((Q^y_{e_2})^x_{e_1}) \wedge (e_2 \text{ évaluable}) \wedge (e_1 \text{ évaluable})*/ \\ &= /*((Q^y_{e_2})^x_{e_1})*/ \qquad \text{(en général, on ne garde que cela !)}\end{aligned}$$

◇ Exercice **1** : Calculer les pfp suivantes :
- $\text{pfp}[i = i + 1, /*i \leq 1*/]$
- $\text{pfp}[i = i + 2; j = j - 2, /*i + j = 0*/]$
- $\text{pfp}[i = i + 1; j = j - 1, /*i \times j = 0*/]$
- $\text{pfp}[z = z \times j; i = i - 1, /*z \times j^i = c*/]$
- $\text{pfp}[x = x - y; y = y - x, /*x + y = c*/]$
- $\text{pfp}[x = x - y; y = y + x, /*x + y = c*/]$

◇ Exercice **2** : Vérifier le programme suivant :

$/*(x = A) \wedge (y = B) \wedge (z = C)*/$
$x = x + y + z;$
$z = x - y - z;$
$y = x - y - z;$
$x = x - y - z;$
$/*(x = B) \wedge (y = C) \wedge (z = A)*/$

◇ Exercice **3** : Indiquer les programmes qui vérifient les spécifications données sachant que $0! = 1$ et $\forall i \in N^*, i! = i \times (i - 1)!$
- Programme 1 :

  $/*f = i!*/$
  $i = i + 1;$
  $f = f \times i;$
  $/*f = i!*/$

- Programme 2 :

  $/*f = i!*/$
  $f = f \times (i + 1);$
  $i = i + 1;$
  $/*f = i!*/$

- Programme 3 :

  $/*f = i!*/$
  $f = f \times i;$
  $i = i + 1;$

$/*f = i!*/$

3 Cas d'une sélection

La règle utilisée est la suivante :

$$\textbf{pfp}[\textbf{if}(B)\{A_1\}\ \textbf{else}\ \{A_2\}, /*PS*/] = (B \longrightarrow \textbf{pfp}[A_1, /*PS*/]) \wedge (\neg B \longrightarrow \textbf{pfp}[A_2, /*PS*/])$$

Cas particulier d'une sélection sans partie *else* : $\textbf{pfp}[\textbf{if}\ (B)\{A_1\}, /*Q*/] = (B \longrightarrow \textbf{pfp}[A_1, /*Q*/]) \wedge (\neg B \longrightarrow Q)$
Pour prouver :
$/*\ PE\ */$ if$(B)\{A_1\}$ else $\{A_2\}$ $/*\ PS\ */$
on prouvera successivement les deux implications suivantes :
$(PE \wedge B) \longrightarrow \text{pfp}[A_1, /*PS*/])$
et
$(PE \wedge \neg B) \longrightarrow \text{pfp}[A_2, /*PS*/])$

◇ Exercice **4** : Calculer les pfp suivantes :

1. pfp$[if\ (x >= y)z = x; elsez = y;\ , /*z = \max(x, y)*/]$
2. pfp$[if\ (x >= y)z = x; elsez = y;\ , /*z = y - 1*/]$
3. pfp$[if\ (x > y)x = y; elsey = x;\ , /*\ x = y\ */]$
4. pfp$[if\ (x \geq y)\{z = x; x = y; y = z\}\ , /*\ x = y\ */]$
5. pfp$\left[\begin{array}{ll} if\ (x > y) & \{if\ (x\ \%2 == 0)\ \{x = x - 2;\}\} \\ else\ y = y - 1; & , /*\ y - 2 < x\ */ \end{array}\right]$

4 Cas d'une répétition

C'est un cas particulier car il faut passer par 5 étapes permettant de réaliser les preuves de correction partielle et de terminaison (c'est effectivement le seul cas où l'on pourrait avoir un programme qui ne se termine pas !).
Soit la spécification suivante :

```
/* PE */
initialisation de boucle ;
/* invariant de boucle (INV) */
while (C)
    {
    /* INV ∧ C */
    corps de boucle ;
    /* INV */
    }
/* INV ∧¬ C */      /* PS */
```

Les étapes à suivre pour faire la preuve de correction partielle sont :

1. **/* PE boucle */ $\longrightarrow$ pfp(initialisation boucle, /* INV */)**
2. **/* INV $\wedge$ C */ $\longrightarrow$ pfp(corps de la boucle, /* INV */)**
3. **/* INV $\wedge$ $\neg C$ */ $\longrightarrow$ /* PS boucle*/**

aprés développement et preuve partielle de boucle (étapes 1, 2 et 3) , pour la preuve de terminaison, on identifie une fonction f appelée **variante** à valeurs entières > 0 telle que :

4. **/* INV $\wedge$ C */ $\longrightarrow$ /* $f > 0$ */ (ce qui signifie que f est strictement positive à l'entrée de la boucle),**
5. **/* $(T = f)\wedge$ INV $\wedge$ C */ $\longrightarrow$ pfp(corps, /* $T > f$ */) (ce qui signifie que f est strictement décroissante).**

$\diamond$ Exercice **5** : On propose des programmes qui prétendent calculer, dans la variable p, la valeur en X d'un polynôme de degré $N \geq 0$. Les coefficients de ce polynôme sont des entiers mémorisés dans un tableau $A[0..N]$. Le résultat de ce programme doit satisfaire la propriété :

$$\mathbf{R} : \{p = \Sigma_{K=0}^{N} A[K]X^K\}$$

Le calcul de cette somme, nécessitant une répétition, nous a conduit à proposer l'invariant suivant :

INV : $\{(0 \leq i \leq N) \bigwedge (p = \Sigma_{K=0}^{i} A[K]X^K) \bigwedge (y = X^i)\}$

1. Indiquer, en justifiant votre réponse, parmi les deux séquences suivantes, celles qui satisfont l'invariant **INV :**

   - init0 : $i = 0; p = 0; y = 1$
   - init1 : $i = 0; p = A[0]; y = 1$

2. L'invariant étant initialement vérifié, indiquer avec une preuve, si, dans le cas où les boucles suivantes se terminent, alors elles se terminent en satisfaisant la propriété **R** :

   - portion boucle_1 :

     while $(i <= N)$
     $\{$ $i = i + 1$;
     $p = p + (A[i-1] \times y)$;
     $y = y \times X$;
     $\}$

   - portion boucle_2 :

     while $(i \neq N)$
     $\{$
     $i = i + 1$;
     $p = p + (A[i-1] \times y)$;
     $y = y \times X$;
     $\}$

– portion boucle_3 :

while $(i \neq N)$

$\{$

$i = i + 1\,;$

$p = p + (A[i] \times y \times X)\,;$

$y = y \times X\,;$

$\}$

3. Pour la portion de boucle_3, donner une variante et faire la preuve de terminaison.