Preuves de programme
Corrigé

Fabrice Rossi

1er février 2012

1 Rappel sur les règles d'inférence

On rappelle les règles suivantes sur les triplets de Hoare :

- **affectation**
  Si y est une nouvelle variable n'apparaissant ni dans le prédicat P ni dans l'expression E, alors le triplet suivant est vrai

$$\langle P \rangle \texttt{ x=E } \langle \exists y, P[x \leftarrow y] \wedge x = E[x \leftarrow y] \rangle ,$$

  où $P[x \leftarrow y]$ désigne le prédicat P dans lequel toutes les occurrences de x ont été remplacées par y (même chose pour $E[x \leftarrow y]$).
- **séquence**
  Si $\{P\}$ `prog1` $\{Q\}$ et $\{Q\}$ `prog2` $\{R\}$ sont vrais, alors $\{P\}$ `prog1; prog2` $\{R\}$ est vrai (cela fonctionne aussi pour la correction totale).
- **sélection**
  Si $\{P \wedge b\}$ `prog1` $\{Q\}$ et $\{P \wedge \neg b\}$ `prog2` $\{Q\}$ sont vrais, alors

$$\{P\} \texttt{ if (}b\texttt{) prog1 else prog2 end if } \{Q\}$$

  est vrai (cela fonctionne aussi pour la correction totale).
- **boucle**
  Si $\{I \wedge b\}$ `prog` $\{I\}$ est vrai, alors

$$\{I\} \texttt{ while(}b\texttt{) prog end while } \{I \wedge \neg b\}$$

  est vrai.
- **boucle avec variant**
  Soit V une expression à valeur entière et I un prédicat tels que le triplet suivant soit vrai

$$\langle I \wedge b \wedge (V = n) \rangle \texttt{ prog } \langle I \wedge (V < n) \rangle .$$

  Si on suppose en outre que $I \Rightarrow V \geq 0$, alors le triplet suivant est vrai

$$\langle I \rangle \texttt{ while(}b\texttt{) prog end while } \langle I \wedge \neg b \rangle .$$

2 Maximum d'un tableau

On considère le programme suivant :

prog

```
1  i = 0
2  x = tab[0]
3  j = 1
4  while(j < tab.length)
5    if(tab[j] > x)
6      x = tab[j]
7      i = j
8    end if
9    j = j + 1
10 end while
```

2.1 Début du programme

On considère la précondition $P = tab.length > 0$. D'après les règles affectation et séquence, le triplet suivant est clairement vrai :

$\langle tab.length > 0 \rangle$
`i = 0`
`x = tab[0]`
`j = 1`
$\langle tab.length > 0 \wedge i = 0 \wedge x = tab[0] \wedge j = 1 \rangle$

2.2 Traitement de la boucle

Pour étudier la boucle, on introduit le prédicat I donné par

$$I = \left(x = \max_{0 \leq k \leq j-1} tab[k] \right) \wedge (j \leq tab.length),$$

ainsi que l'expression entière $V = tab.length - j$. Comme I vrai implique que $j \leq tab.length$, $I \Rightarrow V \geq 0$. Le couple (I, V) est donc un bon candidat pour être respectivement les invariant et variant de la boucle while du programme. On appelle alors `corps de boucle` les lignes 5 à 9 du programme étudié et on analyse le triplet

$$\langle I \wedge (j < tab.length) \wedge (V = n) \rangle \texttt{ corps de boucle } \langle I \wedge (V < n) \rangle .$$

2.2.1 Traitement de la sélection

Pour étudier ce triplet, il faut d'abord analyser la sélection. On remarque que si I est vrai, alors, par définition, x contient le plus grand élément parmi les cases 0 à $j-1$ du tableau. Or la condition du `if` (ligne 5) compare x à $tab[j]$: on voit que l'objectif est du `if` est de mettre dans x la valeur contenue dans $tab[j]$ si cette valeur est plus grande celles contenues dans les cases 0 à $j-1$. Il est donc naturel de supposer qu'après l'exécution de la sélection, on aura $x = \max_{0 \leq k \leq j} tab[k]$ (attention au changement d'indice maximum).

On note

$$P = I \wedge (j < tab.length) \wedge (V = n).$$

Pour appliquer la règle de sélection, on doit étudier les deux alternatives de la sélection, à savoir $P \wedge (tab[j] > x)$ et $P \wedge \neg(tab[j] > x)$. Commençons par le `else` qui est vide ici, et considérons donc $P \wedge \neg(tab[j] > x)$, soit en fait

$$\left(x = \max_{0 \leq k \leq j-1} tab[k] \right) \wedge (j \leq tab.length) \wedge (j < tab.length) \wedge (V = n) \wedge \neg(tab[j] > x).$$

Ce prédicat est clairement équivalent à

$$\left(x = \max_{0 \leq k \leq j-1} tab[k]\right) \wedge (j < tab.length) \wedge (V = n) \wedge tab[j] \leq x,$$

qui implique

$$tab[j] \leq \max_{0 \leq k \leq j-1} tab[k],$$

et donc

$$\max_{0 \leq k \leq j-1} tab[k] = \max_{0 \leq k \leq j} tab[k] = x.$$

On a donc montré que le triplet suivant est vrai

$$\langle P \wedge \neg(tab[j] > x) \rangle \ \emptyset \ \left\langle \left(x = \max_{0 \leq k \leq j} tab[k]\right) \wedge (j < tab.length) \wedge (V = n) \right\rangle, \tag{1}$$

car la postcondition est impliquée par la précondition et qu'il n'y a pas de programme.

Remarque 2.1
Techniquement, il s'agit d'une application de la **règle d'implication** qui est évidente mais qu'on peut rappeler ici.

Si $P \Rightarrow P'$, $Q' \Rightarrow Q$ et que le triplet $\{P'\}$ `prog` $\{Q'\}$ est vrai, alors le triplet $\{P\}$ `prog` $\{Q\}$ est vrai (ceci fonctionne aussi pour la correction totale).

On traite de façon similaire (mais plus complexe) la branche `if` de la sélection. On considère $P \wedge (tab[j] > x)$. En appliquant la règle pour l'affectation, on obtient que le triplet suivant est vrai :

$$\langle P \wedge (tab[j] > x) \rangle \ \texttt{x=tab[j]} \ \langle \exists y, P[x \leftarrow y] \wedge (tab[j] > y) \wedge x = tab[j] \rangle .$$

Or,

$$P[x \leftarrow y] = \left(y = \max_{0 \leq k \leq j-1} tab[k]\right) \wedge (j < tab.length) \wedge (V = n),$$

donc la postcondition du triplet précédent est équivalente à

$$\left(y = \max_{0 \leq k \leq j-1} tab[k]\right) \wedge (j < tab.length) \wedge (V = n) \wedge (tab[j] > y) \wedge x = tab[j].$$

Or, $(tab[j] > y) \wedge (y = \max_{0 \leq k \leq j-1} tab[k])$ implique clairement que $tab[j] = \max_{0 \leq k \leq j} tab[k]$ et donc la postcondition implique

$$\left(x = \max_{0 \leq k \leq j} tab[k]\right) \wedge (j < tab.length) \wedge (V = n).$$

On ne s'occupe pas de `i = j` car la variable `i` n'est pas prise en compte dans nos préconditions et postconditions. On a donc montré que le triplet suivant est vrai :

$$\langle P \wedge (tab[j] > x) \rangle \ \texttt{x=tab[j]; i=j} \ \left\langle \left(x = \max_{0 \leq k \leq j} tab[k]\right) \wedge (j < tab.length) \wedge (V = n) \right\rangle. \tag{2}$$

En appliquant les règles sur la sélection triplets (1) et (2), on en déduit donc que le triplet suivant est vrai :

$\langle I \wedge (j < tab.length) \wedge (V = n) \rangle$

```
if(tab[j] > x)
 x = tab[j]
 i = j
end if
```

$$\left\langle \left(x = \max_{0 \leq k \leq j} tab[k]\right) \wedge (j < tab.length) \wedge (V = n) \right\rangle$$

2.2.2 Étude de l'invariant et du variant

On appelle Q la postcondition démontrée au dessus. D'après les règles d'affectation, le triplet suivant est vrai

$$\langle Q \rangle \texttt{ j = j + 1 } \langle \exists y, Q[j \leftarrow y] \wedge j = y + 1 \rangle \,,$$

Or (attention à V), on a

$$Q[j \leftarrow y] = \left(x = \max_{0 \leq k \leq y} tab[k] \right) \wedge (y < tab.length) \wedge (tab.length - y = n).$$

Comme $y = j - 1$, la postcondition du triplet précédent est donc équivalente à

$$R = \left(x = \max_{0 \leq k \leq j-1} tab[k] \right) \wedge (j - 1 < tab.length) \wedge (tab.length - j + 1 = n).$$

Comme j est entier, $j - 1 < tab.length$ est équivalent à $j \leq tab.length$. On en déduit donc que R implique I, ce qui montre que I est un invariant de boucle. On remarque en outre que R implique $V = n - 1$ (car $V = tab.length - j$). On a donc bien $V < n$, comme voulu. En résumé, on a montré que le triplet suivant est vrai :

$$\langle I \wedge (j < tab.length) \wedge (V = n) \rangle \texttt{ corps de boucle } \langle I \wedge (V < n) \rangle$$

3 Correction totale du programme

La section précédente permet de conclure que le triplet suivant est vrai :

$$\langle I \rangle \texttt{ boucle } \langle I \wedge \neg(j < tab.length) \rangle \,,$$

où `boucle` désigne la boucle du programme, c'est-à-dire les lignes 4 à 10. Or nous avons aussi montré que la précondition $tab.length > 0$ conduit à la postcondition $tab.length > 0 \wedge i = 0 \wedge x = tab[0] \wedge j = 1$ après l'exécution des lignes 1 à 3. Or, il est clair que $x = tab[0] \wedge j = 1$ est équivalent à $x = \max_{0 \leq k \leq j-1} tab[k] \wedge j = 1$. De plus, comme $tab.length$ est entier, $tab.length > 0$ est équivalent $tab.length \geq 1$. Le triplet suivant est donc vrai

$$\begin{array}{l} \langle tab.length > 0 \rangle \\ \texttt{i = 0} \\ \texttt{x = tab[0]} \\ \texttt{j = 1} \\ \langle I \rangle \,, \end{array}$$

et donc le triplet suivant est vrai

$$\langle tab.length > 0 \rangle \texttt{ prog } \langle I \wedge \neg(j < tab.length) \rangle \,,$$

On remarque alors que $I \wedge \neg(j < tab.length)$ implique $j = tab.length$ (car j est entier) et donc que $I \wedge \neg(j < tab.length)$ implique

$$x = \max_{0 \leq k \leq tab.length - 1} tab[k],$$

ce qui achève de montrer la correction (totale) du programme (comme dans la section précédente, on applique la règle d'implication).

4 Compléments d'analyse

4.1 Accès corrects

Le programme accède au tableau *tab* aux lignes 2, 5 et 6. On cherche à montrer que les accès sont toujours corrects, c'est-à-dire que pour toute expression faisant apparaître $tab[k]$, on a $k \geq 0 \wedge k < tab.length$.

La ligne 2 se traite facilement. On utilise en effet $P = tab.length > 0$ comme précondition du programme. Il est alors clair que si P est vraie, $tab[0]$ est bien défini et donc que l'accès est correct.

Pour traiter les lignes 5 et 6, il faut montrer qu'au moment de l'exécution de ces lignes, le prédicat $Q = j \geq 0 \wedge j < tab.length$ est vrai. Il est facile de montrer que $j \leq 1$ à partir de la ligne 4. On considère en effet l'invariant de boucle $J = j \leq 1$. Comme la seule modification de j effectuée dans `corps de boucle` est `j = j + 1`, la correction du triplet

$$\langle J \wedge (j < tab.length) \rangle \texttt{ corps de boucle } \langle J \rangle ,$$

revient à celle de $\langle J \wedge (j < tab.length) \rangle$ `j = j + 1` $\langle J \rangle$ qui est évidente. Formellement, il suffit d'appliquer la règle d'affectation qui assure la correction de

$$\langle j \leq 1 \wedge (j < tab.length) \rangle \texttt{ j = j + 1 } \langle \exists y, y \leq 1 \wedge y < tab.length \wedge j = y + 1 \rangle ,$$

donc la postcondition implique clairement $j \geq 1$.

La condition $j < tab.length$ est aussi facile à vérifier car c'est exactement la condition de la boucle. Comme les lignes 5 et 6 sont exécutées avant la modification de `j` à la ligne 9, il est clair que la condition de la boucle est encore valide à ce moment.

4.2 Position du maximum

Le programme maintient une variable `i` qui indique la position du maximum. Plus précisément, on a après l'exécution du programme

$$i = \min \left\{ 0 \leq k < tab.length \middle| tab[k] = \max_{0 \leq l < tab.length} tab[l] \right\} .$$

On montre cette propriété par une technique classique d'invariant. Appelons S le prédicat suivant

$$S = \left(i = \min \left\{ 0 \leq k \leq j-1 \middle| tab[k] = \max_{0 \leq l \leq j-1} tab[l] \right\} \right) .$$

On remarque que $I \wedge S$ est équivalent à

$$I \wedge \left(i = \min \left\{ 0 \leq k \leq j-1 \middle| tab[k] = x \right\} \right) .$$

On doit alors analyser l'effet de `corps de boucle` sur ce prédicat. Comme dans l'analyse de la section 2.2, on remarque que si $tab[k] \leq x$, S reste vrai après l'« exécution »de la sélection (qui ne fait rien, puisqu'il n'y a pas de `else`).

Plus précisément, $tab[k] \leq x$ est équivalent à $tab[k] < x \vee tab[k] = x$. Si $tab[k] < x$ alors

$$\left\{ 0 \leq k \leq j-1 \middle| tab[k] = x \right\} = \left\{ 0 \leq k \leq j \middle| tab[k] = x \right\} ,$$

et donc

$$\left(i = \min \left\{ 0 \leq k \leq j-1 \middle| tab[k] = x \right\} \right) \Rightarrow \left(i = \min \left\{ 0 \leq k \leq j \middle| tab[k] = x \right\} \right) .$$

Si $tab[k] = x$, alors

$$\left\{0 \leq k \leq j \middle| tab[k] = x\right\} = \left\{0 \leq k \leq j-1 \middle| tab[k] = x\right\} \cup \{k\},$$

et donc

$$\left(i = \min\left\{0 \leq k \leq j-1 \middle| tab[k] = x\right\}\right) \Rightarrow \left(i = \min\left\{0 \leq k \leq j \middle| tab[k] = x\right\}\right).$$

De ce fait,

$$\left(i = \min\left\{0 \leq k \leq j-1 \middle| tab[k] = x\right\}\right) \wedge (tab[k] \leq x) \Rightarrow \left(i = \min\left\{0 \leq k \leq j \middle| tab[k] = x\right\}\right).$$

On montre de la même façon que si $tab[k] > x$, l'exécution des lignes 6 et 7 du programme conduit à la même propriété. En appliquant les règles sur les sélections, on conclut que le triplet suivant est vrai :

$\langle I \wedge S \rangle$

```
if(tab[j] > x)
 x = tab[j]
 i = j
end if
```

$$\left\langle \left(x = \max_{0 \leq k \leq j} tab[k]\right) \wedge \left(i = \min\left\{0 \leq k \leq j \middle| tab[k] = x\right\}\right) \right\rangle$$

Intuitivement, on a donc fait avancer l'invariant de $j-1$ à j. Le reste de la preuve se fait de façon quasi-identique à ce qui a été fait précédemment.