UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Correction et terminaison

1 Petits exercices « à la main »

Correction

Démontrer la correction de l'algorithme suivant :

1. Écrire une spécification formelle du problème traité.

   Corrigé Précondition : $n \geq 0$ entier
   Postcondition : $F = n!$.

2. Déterminer un invariant de boucle judicieux.

   Corrigé On démontrera qu'en début d'itération on a $F \times i! = n!$.

3. Établir que cette propriété est effectivement un invariant et conclure.

   Corrigé Preuve de l'invariant : Si l'invariant $F \times i! = n!$ est vérifié au début d'une itération, alors à la fin de cette itération $F' \times i'! = (F \times i) \times (i-1)! = F \times i \times (i-1)! = F \times i! = n!$
   De plus, à l'entrée dans la boucle $F = 1$ et $i = n$ donc l'invariant est vérifié.
   On en déduit qu'en sortie de boucle $n! = F \times i! = F \times 0! = F$.

```
FACTO(n)
  Data : un entier n ≥ 0
  Result : la factorielle de n
  F := 1
  i := n
  while i > 0 do
  |  F := F × i
  |  i := i − 1
  return F
```

Terminaison

Établir la terminaison de l'algorithme suivant :

```
RAC(n)
  Data : un entier n ≥ 0
  Result : la racine carrée de n (arrondie à l'entier inférieur)
  c := 0
  s := 1
  while s ≤ n do
  |  c := c + 1
  |  s := s + 2 ∗ c + 1
  return c
```

Corrigé Il suffit d'exhiber un variant de boucle. La quantité $n - s$:
- est positive (cf condition du while)
- décroît strictement à chaque itération car n ne varie pas et s augmente de $2c + 1$.

On fera remarquer que la dernière affirmation demande cependant de vérifier que la valeur de c est toujours positive (invariant facile à démontrer).
On peut aussi, de façon équivalente, démontrer que la valeur de s augmente strictement à chaque itération et qu'elle est bornée par n, qui lui est constant.

2 Logique de Hoare

On rappelle les règles de la logique de Hoare :

$$\frac{\{P\}\ \texttt{I}\ \{Q\} \qquad \{Q\}\ \texttt{J}\ \{R\}}{\{P\}\ \texttt{I ; J}\ \{R\}} \qquad \frac{\{P \wedge C\}\ \texttt{I}\ \{Q\} \qquad \{P \wedge \neg C\}\ \texttt{J}\ \{Q\}}{\{P\}\ \texttt{if C then I else J}\ \{Q\}} \qquad \frac{\{P \wedge C\}\ \texttt{I}\ \{P\}}{\{P\}\ \texttt{while C do I}\ \{P \wedge \neg C\}}$$

$$\frac{}{\{Q[\texttt{x} \leftarrow \texttt{E}]\}\ \texttt{x := E}\ \{Q\}} \qquad \frac{P \Rightarrow P' \qquad \{P'\}\ \texttt{I}\ \{Q\}}{\{P\}\ \texttt{I}\ \{Q\}} \qquad \frac{\{P\}\ \texttt{I}\ \{Q\} \qquad Q \Rightarrow Q'}{\{P\}\ \texttt{I}\ \{Q'\}} \qquad \frac{}{\{P\}\ \texttt{skip}\ \{P\}}$$

Échauffement

Construire un arbre de preuve pour chacun des triplets suivants :
- $\{\texttt{x} = \texttt{y}\}$ `x := x + y` $\{\texttt{x} = 2 * \texttt{y}\}$
- $\{true\}$ `while x < 0 do x := x + 1` $\{\texttt{x} \geq 0\}$
- $\{\texttt{x} \leq 0\}$ `while x < 0 do x := x + 1` $\{\texttt{x} = 0\}$
- $\{\texttt{x} = 42\}$ `while x > 0 do x := x + 1` $\{\texttt{x} \leq 0\}$
- $\{false\}$ `I` $\{Q\}$ quels que soient l'instruction `I` et la postcondition Q
- $\{P\}$ `I` $\{true\}$ quels que soient l'instruction `I` et la précondition P

Construire également une preuve de $\{P\}$ `if C then I` $\{Q\}$ sous les hypothèses suivantes : $\{P \wedge \texttt{C}\}$ `I` $\{Q\}$ et $P \wedge \neg\texttt{C} \Rightarrow Q$.

Corrigé On se contente des arbres de preuve ; pour s'aider on procède en partant de la postcondition.
—

$$\frac{}{\{\texttt{x} + \texttt{y} = 2 * \texttt{y}\}\ \texttt{x := x + y}\ \{\texttt{x} = 2 * \texttt{y}\}}$$

et la précondition est équivalente à $\texttt{x} = \texttt{y}$.
Au pire, on utilise une règle logique :

$$\frac{\texttt{x} = \texttt{y} \ \Rightarrow\ \texttt{x} + \texttt{y} = 2 * \texttt{y} \qquad \{\texttt{x} + \texttt{y} = 2 * \texttt{y}\}\ \texttt{x := x + y}\ \{\texttt{x} = 2 * \texttt{y}\}}{\{\texttt{x} = \texttt{y}\}\ \texttt{x := x + y}\ \{\texttt{x} = 2 * \texttt{y}\}}$$

— La postcondition $\texttt{x} \geq 0$ est exactement la négation de la condition du while : on prend comme invariant de la boucle $P = true$.

$$\frac{\{true \wedge \texttt{x} < 0\}\ \texttt{x := x + 1}\ \{true\}}{\{true\}\ \texttt{while x < 0 do x := x + 1}\ \{true \wedge \texttt{x} \geq 0\}}$$

cf plus bas pour la preuve de $\{true \wedge \texttt{x} < 0\}$ `x := x + 1` $\{true\}$.
— Ici la précondition $\texttt{x} \leq 0$ est un invariant adapté, on le reporte dans la postcondition avec la négation de la condition du while, et on obtient (presque) directement la postcondition recherchée :

$$\dfrac{\dfrac{\dfrac{\texttt{x} \leq 0 \wedge \texttt{x} < 0 \Rightarrow \texttt{x} + 1 \leq 0 \qquad \dfrac{}{\{\texttt{x} + 1 \leq 0\}\ \texttt{x := x + 1}\ \{\texttt{x} \leq 0\}}}{\{\texttt{x} \leq 0 \wedge \texttt{x} < 0\}\ \texttt{x := x + 1}\ \{\texttt{x} \leq 0\}}}{\{\texttt{x} \leq 0\}\ \texttt{while x < 0 do x := x + 1}\ \{\texttt{x} \leq 0 \wedge \texttt{x} \geq 0\}} \qquad \texttt{x} \leq 0 \wedge \texttt{x} \geq 0 \Rightarrow \texttt{x} = 0}{\{\texttt{x} \leq 0\}\ \texttt{while x < 0 do x := x + 1}\ \{\texttt{x} = 0\}}$$

On ne s'attarde pas sur la preuve des deux implications...

— Pour cette preuve et pour la suivante, on admettra que, pour tout programme `I` :
  — pour toute précondition P il existe une postcondition Q telle que $\{P\}$ `I` $\{Q\}$
  — pour toute postcondition Q il existe une précondition P telle que $\{P\}$ `I` $\{Q\}$
  (ce qui se démontre par exemple par induction structurelle sur `I`).
  On conclut alors :

$$\frac{false \Rightarrow P \qquad \{P\}\ \texttt{I}\ \{Q\}}{\{false\}\ \texttt{I}\ \{Q\}}$$

— Même principe :

$$\frac{\{P\}\ \texttt{I}\ \{Q\} \qquad Q \Rightarrow true}{\{P\}\ \texttt{I}\ \{true\}}$$

  Pour ces deux dernières preuves, il n'est pas inutile de se remémorer la table de vérité de $\Rightarrow$.
— On réécrit `if C then I` avec un `skip` :

$$\frac{\{P \wedge C\}\ \texttt{I}\ \{Q\} \qquad \dfrac{P \wedge \neg C \Rightarrow Q \qquad \dfrac{}{\{Q\}\ \texttt{skip}\ \{Q\}}}{\{P \wedge \neg C\}\ \texttt{skip}\ \{Q\}}}{\{P\}\ \texttt{if C then I else skip}\ \{Q\}}$$

Exercice 1

1. Spécifier un programme qui calcule le maximum de deux variables `x` et `y`.

   Corrigé Pas de précondition ($P = true$).
   On cherche à prouver la postcondition $Q = \texttt{m} \geq \texttt{x} \wedge \texttt{m} \geq \texttt{y} \wedge (\texttt{m} = \texttt{x} \vee \texttt{m} = \texttt{y})$, où `m` est la valeur renvoyée par l'algorithme.
2. Écrire ce programme.

   Corrigé `if x>y then m:=x else m:=y`
3. Démontrer sa correction en logique de Hoare.

   Corrigé La seule difficulté est d'effectuer correctement la substitution dans Q :
   — $Q[\texttt{m} \leftarrow \texttt{x}] \quad = \quad \texttt{x} \geq \texttt{x} \wedge \texttt{x} \geq \texttt{y} \wedge (\texttt{x} = \texttt{x} \vee \texttt{x} = \texttt{y})$
   — $Q[\texttt{m} \leftarrow \texttt{y}] \quad = \quad \texttt{y} \geq \texttt{x} \wedge \texttt{y} \geq \texttt{y} \wedge (\texttt{y} = \texttt{x} \vee \texttt{y} = \texttt{y})$
   Dans les deux cas la disjonction finale est trivialement vérifiée ainsi qu'une des inégalités, on peut les simplifier pour se ramener à une unique inégalité.

$$\frac{\dfrac{\texttt{x} > \texttt{y} \Rightarrow \texttt{x} \geq \texttt{y} \qquad \{Q[\texttt{m} \leftarrow \texttt{x}]\}\ \texttt{m:=x}\ \{Q\}}{\{x > y\}\ \texttt{m:=x}\ \{Q\}} \qquad \dfrac{\texttt{x} \leq \texttt{y} \Rightarrow \texttt{y} \geq \texttt{x} \qquad \{Q[\texttt{m} \leftarrow \texttt{y}]\}\ \texttt{m:=y}\ \{Q\}}{\{x \leq y\}\ \texttt{m:=y}\ \{Q\}}}{\{true\}\ \texttt{if x>y then m:=x else m:=y}\ \{Q\}}$$

Exercice 2

Soit `I` l'extrait de programme :

```
si a est pair alors
    a := a/2
sinon
    a := 2*a +1
```

Trouver une précondition à la postcondition $Q = a\ est\ impair$.

Corrigé Des règles de la conditionnelle et de l'affectation on déduit facilement qu'une précondition à la postcondition $Q = a\ est\ impair$ est :
$a\ pair \wedge a/2\ impair \vee a\ impair \wedge 2 * a + 1\ impair$
On profite du fait que $2a + 1$ est toujours impair et on simplifie pour obtenir « *a n'est pas multiple de 4* ».
Remarque : on arrive au même résultat en écrivant la précondition
$(a\ pair \Rightarrow a/2\ impair) \wedge (a\ impair \Rightarrow 2 * a + 1\ impair)$

Exercice 3

On reprend le programme RAC de la page précédente.
On cherche à prouver la spécification suivante :

- La précondition est $n \geq 0$.
- La postcondition est $c \geq 0\ \wedge\ c^2 \leq n < (c+1)^2$

1. Démontrer en logique de Hoare que $c \geq 0\ \wedge\ c^2 \leq n\ \wedge\ s = (c+1)^2$ est un invariant de la boucle.

   Corrigé La preuve de l'invariant (notons-le Inv) se résume à :

$$\dfrac{\dfrac{\vdots}{\{Inv \wedge s \leq n\}\ \texttt{c:=c+1}\ \{c \geq 0 \wedge c^2 \leq n \wedge s = c^2\}} \qquad \dfrac{}{\{c \geq 0 \wedge c^2 \leq n \wedge s = c^2\}\ \texttt{s:=s+2*c+1}\ \{Inv\}}}{\{Inv \wedge s \leq n\}\ \texttt{c:=c+1 ; s:=s+2*c+1}\ \{Inv\}}$$

   En effet, pour la sous-preuve de droite, en substituant s par $s + 2c + 1$ dans l'invariant, on obtient $s + 2c + 1 = (c+1)^2$ ce qui est équivalent à $s = c^2$.
   Pour la sous-preuve de gauche, on est ramené à démontrer l'implication

$$c \geq 0 \wedge c^2 \leq n \wedge s = (c+1)^2 \wedge s \leq n \Rightarrow c + 1 \geq 0 \wedge (c+1)^2 \leq n \wedge s = (c+1)^2$$

   On montre facilement que :
   — $c \geq 0 \Rightarrow c + 1 \geq 0$ d'une part
   — $s = (c+1)^2 \wedge s \leq n \Rightarrow (c+1)^2 \leq n$ d'autre part
   ce qui suffit à conclure.

2. En déduire que le programme est correct.

   Corrigé En entrée de boucle, l'invariant est trivialement vérifié par $c := 0\ ;\ s := 1$.
   En sortie de boucle, puisque $s > n$ et $s = (c+1)^2$, immédiatement $n < (c+1)^2$. Le reste de la postcondition est directement fourni par l'invariant.

Exercice bonus

Soit I l'extrait de programme :

```
i := 1
j := n
tantque i<j faire
      si t(i)<=t(j) alors
              i := i+1
      sinon
              j := j-1
resultat := t(i)
```

Que fait ce programme ? Le prouver.