TD 4 : Logique de Hoare

Matthieu Sozeau (matthieu.sozeau@inria.fr)

October 20, 2014

Ce TD porte sur la preuve de programmes impératifs en utilisant la logique de Hoare. Les notes de cours et corrigés des TDs précédents sont disponibles sur la page du cours:

www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014.fr.html

On rappelle que les règles d'inférence de la logique de Hoare sont données par :

$$\{P\}\ \texttt{skip}\ \{P\} \qquad \{P[exp/x]\}\ x := exp\ \{P\}$$

$$\frac{\{P\}\ C\ \{Q\} \qquad \{Q\}\ D\ \{R\}}{\{P\}\ C; D\ \{R\}}$$

$$\frac{\{E = \texttt{true} \wedge P\}\ C\ \{Q\} \qquad \{E = \texttt{false} \wedge P\}\ D\ \{Q\}}{\{P\}\ \texttt{if}\ E\ \texttt{then}\ C\ \texttt{else}\ D\ \{Q\}}$$

$$\frac{\{E = \texttt{true} \wedge I\}\ C\ \{I\}}{\{I\}\ \texttt{while}\ E\ \texttt{do}\ C\ \texttt{done}\ \{E = \texttt{false} \wedge I\}}$$

$$\frac{P^{\emptyset} \Rightarrow P \qquad \{P\}\ C\ \{Q\} \qquad Q \Rightarrow Q^{\emptyset}}{\{P^{\emptyset}\}\ C\ \{Q^{\emptyset}\}}$$

(où C, D désignent des commandes, E, V des expressions sans effets de bord).

Triplets

Exercice 1. *Dire pour chacun des triplets suivants s'ils sont derivables ou non. S'ils sont derivables, donner les regles d'inference necessaires et les raisonnement logiques utiles.*

- $\{x = 2\}\ x := 3\ \{x = 3\}$
- $\{x = 2\}\ x := x + 1\ \{x = 3\}$
- $\{y = 2\}\ x := y\ \{x = 2\}$
- $\{y > 0\}\ x := y; y = \quad 1\ \{x > 0\}$
- $\{y > 0\}\ \texttt{if}\ \ y > 0\ \ \texttt{then}\ x := y\ \ \texttt{else}\ x := \quad y\ \{x > 0\}$
- $\{\texttt{>}\}\ \texttt{if}\ \ y > 0\ \ \texttt{then}\ x := y\ \ \texttt{else}\ x := \quad y\ \{x > 0\}$

Solution

- Assignement: $\{3 = 3\}\ x := 3\ \{x = 3\}$ et implication $x = 2 \Rightarrow 3 = 3$.
- Assignement $\{x + 1 = 3\}\ x := x + 1\ \{x = 3\}$ et implication $x = 2 \Rightarrow x + 1 = 3$.
- Assignement $\{y = 2\}\ x := y\ \{x = 2\}$
- Par assignement on a: $\{y > 0\}\ x := y\ \{x > 0\}$ et $\{x > 0 \wedge \quad 1 = \quad 1\}\ y := \quad 1\ \{x > 0 \wedge y = \quad 1\}$

  On compose ces triplets en affaiblissant le deuxième puisque $x > 0 \Rightarrow x > 0 \wedge \quad 1 = \quad 1$ et $x > 0 \wedge y = \quad 1 \Rightarrow x > 0$.

  $\{y > 0\}\ x := y; y := \quad 1\ \{x > 0\}$
- On utilise la règle de la conditionelle pour se ramener à deux cas:

  $\{y > 0 \wedge y > 0\}\ x := y\ \{x > 0\}$ et $\{y > 0 \wedge \neg(y > 0)\}\ x := \quad y\ \{x > 0\}$.

  Le premier est valide par affaiblissement et assignement. Le deuxième est valide puisque par assignement on a un triplet: $\{\quad y > 0\}\ x := \quad y\ \{x > 0\}$. Or $y > 0 \wedge \neg(y > 0) \Rightarrow ?$. Par transitivité avec $? \Rightarrow \quad y > 0$ ($? \Rightarrow \phi$ est toujours vraie), on obtient:

  $$y > 0 \wedge \neg(y > 0) \Rightarrow \quad y > 0$$

  On peut donc appliquer la règle de conséquence en utilisant cette implication pour la précondition et $x > 0 \Rightarrow x > 0$ pour la postcondition.
- Triplet non démontrable sans condition sur y.

Exercice 2 (Euclide, again). *Trouvez la postcondition et l'invariant de boucle pour la division euclidienne de x par y de nie iterativement puis faites la preuve de sa correction.*

```
assume(x >= 0 /\ y > 0);
a := 0;
b := x;
while b >= y do
  b := b - y;
  a := a + 1;
done;
assert(?)
```

Exercice 3 (Renversement itératif). *Sans vous aider du cours, retrouvez l'invariant de boucle de la version iterative du renversement d'une liste. Veri ez la correction de la fonction en donnant et justi ant les triplets de Hoare pour chaque commande.*

$\rho : List[\star]$;

irev $(\ell : List[\star]) =$
  assume$(true)$;
  $\ell^{\theta} : List[\star]$;
  $\rho := []$; % ρ is the reverse of the treated pre x of ℓ
  $\ell^{\theta} := \ell$; % ℓ^{θ} is the non-treated su x of ℓ
  while $\ell^{\theta} \neq []$ do
    invariant?
    $\rho := head(\ell^{\theta}) :: \rho$;
    $\ell^{\theta} := tail(\ell^{\theta})$;
  assert$(\rho = Rev(\ell))$

Solution

$\rho : List[\star]$;

irev $(\ell : List[\star]) =$
assume$(true)$;
$\ell^\theta : List[\star]$;
$f\ell = \ell g$
$f\ell = rev([])@\ell g$
$\rho := []$; % ρ is the reverse of the treated prefix of ℓ
$f\ell = rev(\rho)@\ell g$
$\ell^\theta := \ell$; % ℓ^θ is the non-treated suffix of ℓ
$f\ell = rev(\rho)@\ell^\theta g$
while $\ell^\theta \neq []$ do
invariant $\ell = rev(\rho)@\ell^\theta$
$f\ell = rev(\rho)@\ell^\theta \wedge \ell^\theta \neq [] g$
$f\ell = rev(\rho)@head(\ell^\theta)\ tail(\ell^\theta)\ ,\ \ell = rev(head(\ell^\theta)\ \rho)@tail(\ell^\theta) g$
$\rho := head(\ell^\theta)\ \rho$;
$f\ell = rev(\rho)@tail(\ell^\theta) g$
$\ell^\theta := tail(\ell^\theta)$;
$f\ell = rev(\rho)@\ell^\theta g$
$f\ell = rev(\rho)@\ell^\theta \wedge \ell^\theta = [] g$
$f\rho = rev(\ell)) g$
assert$(\rho = rev(\ell))$

Exercice 4. (Tableaux) *En logique de Hoare, les tableaux sont manipules a l'aide des deux fonctions suivantes :*

- *La fonction* `access(t, i)` *qui retourne le* i*-eme element du tableau* t*;*
- *La fonction* `store(t, i, v)` *qui retourne un nouveau tableau ayant les mêmes elements que* t*, sauf le* i*-eme qui est remplace par* v*.*

Ces deux fonctions permettent de de nir la lecture et l'ecriture dans un tableau:[1]

$$t[i] \qquad \texttt{access}(t,i) \qquad \text{et} \qquad t[i] := E \qquad t := \texttt{store}(t,i,E)$$

1. *Quels axiomes est-il raisonnable de supposer sur les fonctions access et store ?*
2. *Montrer la correction du programme suivant (ou x et y sont des variables fra^ches) :*

$$f t[i] = x \wedge t[j] = y g \quad v := t[i];\ t[i] := t[j];\ t[j] := v \quad f t[i] = y \wedge t[j] = x g$$

Solution

1. $f>g\ store(t,i,a)\ f t[i] = a g$ et $f t[i] = a g\ x := access(t,i)\ f x = a g$.
2. Voir exemple swap du cours.

Exercice 5 (Calcul de la racine par addition)**.** *On cherche un programme calculant la racine carre entiere d'un entier.*

[1] On notera que l'opération qui consiste à écrire dans une seule case du tableau est traduite en logique de Hoare par le remplacement du tableau complet.

1. *Donner la speci cation du probleme.*

2. *Implementer une solution na•ve du probleme qui repose sur une methode par incrementation.*

3. *Annoter et prouver votre implementation en utilisant la logique de Hoare.*

4. *On considere l'amelioration du programme ou l'on remplace une multiplication coûteuse par une addition.*

```
r := 0; y := 1; z := 1;
while (y<=n) do
  z := z+2; y := y+z; r := r+1;
done;
```

A l'aide des axiomes d'a ectation, trouver les expressions E_i et S_i a n que les formules suivantes soient valides :

$$\begin{array}{llll}
\{E_1\} & \texttt{z:=1} & \{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \\
\{E_2\} & \texttt{y:=1} & \{y = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad n\} \\
\{E_3\} & \texttt{r:=0} & \{1 = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad n\} \\
\{E_4\} & \texttt{r:=r+1} & \{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \\
\{E_5\} & \texttt{y:=y+z} & \{y = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad n\} \\
\{E_6\} & \texttt{z:=z+2} & \{y+z = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad n\}
\end{array}$$

$$\begin{array}{llll}
\{y = (r+1)^2 \wedge z = 2r+3 \wedge r^2 \quad n\} & \texttt{y:=y+z} & \{S_1\} \\
\{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} & \texttt{z:=z+2} & \{S_2\}
\end{array}$$

5. *A l'aide de la regle de la sequence et de l'a aiblissement, demontrer les theoremes suivants: Init:*

$$\begin{array}{l}
\{n \quad 0\} \\
r := 0;\ y := 1;\ z := 1 \\
\{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\}
\end{array}$$

Loop:

$$\begin{array}{l}
\{y <= n \wedge y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \\
z := z+2;\ y := y+z;\ r := r+1 \\
\{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\}
\end{array}$$

6. *En appliquant la regle d'iteration, montrer la validite de la formule:*

$$\begin{array}{l}
\{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \\
\textit{while}\ y \quad n\ \textit{do}\ z := z+2;\ y := y+z;\ r := r+1\ \textit{done} \\
\{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n \wedge y > n\}
\end{array}$$

7. *En vous aidant des questions precedentes, annoter le programme et prouver sa correction.*

Solution:

1. Le programme racine carré entière prend un argument `n : Nat` et renvoie un entier `r : Nat`

- Pré condition : `{}`
- Post condition : `{r * r <= n < (r+1)*(r+1)}`

2.
```
r := 0;
while ((r+1)*(r+1) <= n) do
  r := r+1;
done;
```

3. On annote le programme en utilisant les règles de séquence, boucle et affectation.

$f I[r \quad 0]g$
`r := 0;`
fIg
`while ((r+1)*(r+1) <= n) do`
$\quad fI \wedge (r+1) \quad (r+1) \quad ng$
`  r := r+1;`
$\quad fIg$
`done;`
$fr \quad r \quad n < (r+1) \quad (r+1)g \overset{①}{,} fI \wedge (r+1) \quad (r+1) > ng$

On propose comme invariant : $I \quad r \quad r \quad n$ et on vérifie bien l'équivalence ①.

En reportant l'invariant dans les annotations et en utilisant la règle d'affectation dans la boucle, on obtient :

$f0 \quad ng$
`r := 0;`
$fr \quad r \quad ng$
`while ((r+1)*(r+1) <= n) do`
$\quad fr \quad r \quad n \wedge (r+1) \quad (r+1) \quad ng \overset{②}{)} f (r+1) \quad (r+1) \quad ng$
`  r := r+1;`
$\quad fr \quad r \quad ng$
`done`
$fr \quad r \quad n < (r+1) \quad (r+1)g \overset{①}{,} fI \wedge (r+1) \quad (r+1) > ng$

On termine la preuve en utilisant la règle Pré et l'implication ②.

4.

$$fE_1g = fy = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad ng \quad \texttt{z:=1} \quad fy = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad ng = fE_0g$$
$$fE_2g = f1 = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad ng \quad \texttt{y:=1} \quad fy = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad ng = fE_1g$$
$$fE_3g = f1 = (0+1)^2 \wedge 1 = 1^2 \wedge 0^2 \quad ng \quad \texttt{r:=0} \quad f1 = (r+1)^2 \wedge 1 = 2r+1 \wedge r^2 \quad ng = fE_2g$$
$$fE_4g = fy = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad ng \quad \texttt{r:=r+1} \quad fy = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad ng = fE_0g$$
$$fE_5g = fy+z = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad ng \quad \texttt{y:=y+z} \quad fy = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad ng = f$$
$$fE_6g = fy+z+2 = (r+2)^2 \wedge z = 2r+1 \wedge (r+1)^2 \quad ng \quad \texttt{z:=z+2} \quad fy+z = (r+2)^2 \wedge z = 2r+3 \wedge (r+1)^2 \quad ng$$

$$fy = (r+1)^2 \wedge z = 2r+3 \wedge r^2 \quad ng \quad \texttt{y:=y+z} \quad fy = (r+1)^2 + 2r+3 \wedge z = 2r+3 \wedge r^2 \quad ng$$
$$fy = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad ng \quad \texttt{z:=z+2} \quad fy = (r+1)^2 \wedge z = 2r+3 \wedge r^2 \quad ng$$

5. • Grâce à la règle de séquence, la règle Pré, et la question précédente on a :

$$\begin{array}{l} \{n \quad 0\} \Rightarrow \{E_3\} \\ r := 0 \\ \{E_2\} \\ y := 1 \\ \{E_1\} \\ z := 1 \\ \{E_0\} = \{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \end{array}$$

• On remarque que la précondition $\{y <= n \wedge y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\}$ permet de déduire :

$$y + z = (r+1)^2 + 2r + 1$$

et donc :

$$y + z = r^2 + 2r + 1 + 2r + 1$$

ce qui se simplifie en :

$$y + z + 2 = (r+2)^2$$

On peut donc vérifier la validité de la formule grâce à la règle de séquence, la règle Pré, et la question 4 :

$$\begin{array}{l} \{y <= n \wedge y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \Rightarrow E_6 \\ z := z + 2 \\ \{E_5\} \\ y := y + z \\ \{E_4\} \\ r := r + 1 \\ \{E_0\} = \{y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n\} \end{array}$$

6. On choisit pour I la formule $y = (r+1)^2 \wedge z = 2r+1 \wedge r^2 \quad n$. On peut directement appliquer la règle de boucle car

• après la boucle on a $\{I \wedge : E\}$ où E est la condition de boucle $(E = y \quad n)$ donc $: E = y > n$.

• On peut annoter l'intérieur de la boucle avec $\{I \wedge E\} C \{I\}$ où $C = z := z+2;\ y := y+z;\ r := r+1$ et ce triplet est valide grâce à la question précédente.

7. Il suffit de remettre toutes les formules au bon endroit dans l'annotation.

```
{n    0}
r:=0; y:=1; z:=1 Question 5
{ y = (r + 1)^2  ^  z = 2r + 1  ^  r^2    n}
while (y<=n) do
  z := z+2; y := y+z; r := r+1;  Question 6
done;
 { y = (r + 1)^2  ^  z = 2r + 1  ^  r^2    n ^  y > n}
④ {r^2    n  <  (r + 1)^2}
```

L'implication ④ est vérifiée en remplaçant y par sa valeur dans la denière partie de la conjonction.