Delphine Longuet, Georges Ouffoué
longuet@lri.fr, ouffoue@lri.fr

TD 5 - Preuve de programmes

22 mars 2017

Exercice 1
Dériver les triplets de Hoare suivants en utilisant les règles d'inférence introduites dans le cours. Rappel : toutes les variables sont des entiers.

1. $\{x \leq 0\}$ `x := x-1` $\{x < 5\}$
2. $\{a = x \wedge b = y\}$ `a := a + b; b := a - 2*b; a := a * b` $\{a = x^2 - y^2\}$
3. $\{i = 8\}$ `WHILE i < 5 DO i := 2*i` $\{i \geq 5\}$

Exercice 2
On considère le programme `Prog` suivant :

```
IF x > y
THEN max := x
ELSE max := y
```

Quelles sont les pré et post-conditions de ce programme ? Démontrer la validité du triplet de Hoare correspondant.

Exercice 3
On considère le programme `Prog` suivant :

```
WHILE y != x DO
   x := x - 1;
   y := y - 2
```

1. Quelles sont les pré et post-conditions de ce programme ?
2. Quel est l'invariant de la boucle ?
3. Démontrer la validité du triplet de Hoare correspondant à ce programme.
4. Donner un variant pour la boucle `WHILE`, c'est-à-dire une expression toujours positive et qui décroît strictement à chaque tour de boucle.

Exercice 4
On veut prouver que le programme suivant effectue la division euclidienne de A par B pour $A \geq 0$ et $B > 0$.

```
Q := 0;
R := A;
WHILE R >= B DO
   R := R - B;
   Q := Q + 1
```

1. Écrire la spécification du programme sous forme de pré et post-conditions.
2. Quel est le triplet de Hoare à prouver ?
3. Trouver un invariant Inv pour la boucle `WHILE`.
4. Dériver le triplet de Hoare :

$$\{A = B \times Q + R \wedge R \geq B\}\ \texttt{R:=R-B; Q:=Q+1}\ \{A = B \times Q + R \wedge R \geq 0\}$$

Indication : trouvez une propriété S qui permette de démontrer le triplet

$$\{S\}\ \texttt{Q:=Q+1}\ \{A = B \times Q + R \wedge R \geq 0\}$$

5. On note `Loop` la boucle `WHILE` du programme. Complétez la preuve précédente afin de dériver le triplet :

$$\{A = B \times Q + R \wedge R \geq 0\}\ \texttt{Loop}\ \{A = B \times Q + R \wedge R \geq 0 \wedge R < B\}$$

6. Dériver le triplet :

$$\{A \geq 0 \wedge B > 0\}\ \texttt{Q:=0; R:=A}\ \{A = B \times Q + R \wedge R \geq 0\}$$

7. Quelle règle faut-il appliquer sur les triplets des questions 5 et 6 pour terminer la preuve du programme ?
8. Donner un variant pour la boucle de ce programme, c'est-à-dire une expression entière dont la valeur est positive et qui décroît à chaque tour de boucle.

Calcul de Hoare

$$\frac{}{\{P\}\ \texttt{SKIP}\ \{P\}}\ \textsf{skip} \qquad \frac{}{\{P[x \mapsto exp]\}\ \texttt{x := exp}\ \{P\}}\ \textsf{aff}$$

$$\frac{\{P \wedge cond\}\ \texttt{ins}_1\ \{Q\} \qquad \{P \wedge \neg cond\}\ \texttt{ins}_2\ \{Q\}}{\{P\}\ \texttt{IF cond THEN ins}_1\ \texttt{ELSE ins}_2\ \{Q\}}\ \textsf{if}$$

$$\frac{\{P \wedge cond\}\ \texttt{ins}\ \{P\}}{\{P\}\ \texttt{WHILE cond DO ins}\ \{P \wedge \neg cond\}}\ \textsf{while}$$

$$\frac{P \Rightarrow P' \qquad \{P'\}\ \texttt{ins}\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ \texttt{ins}\ \{Q\}}\ \textsf{cons}$$

$$\frac{}{\{false\}\ \texttt{ins}\ \{P\}}\ \textsf{falseE} \qquad \frac{\{P\}\ \texttt{ins}_1\ \{Q\} \qquad \{Q\}\ \texttt{ins}_2\ \{R\}}{\{P\}\ \texttt{ins}_1\ \texttt{;}\ \texttt{ins}_2\ \{R\}}\ \textsf{seq}$$

Exercice 5
On considère l'instruction de boucle `DO ins UNTIL cond` dans laquelle la condition de sortie est vérifiée après le corps de la boucle. Celui-ci est donc toujours exécuté au moins une fois.

1. Donner une règle d'inférence pour cette instruction dans la logique de Hoare.
2. Montrer que cette règle peut se déduire des règles habituelles en considérant que la boucle `DO ins UNTIL cond` est équivalente au programme

   `ins ; WHILE not cond DO ins`

3. Réciproquement, montrer que la nouvelle règle permet de retrouver la règle classique du `WHILE`, si on traduit l'instruction `WHILE cond DO ins` par le programme

   `IF cond THEN DO ins UNTIL not cond ELSE SKIP`

Exercice 6
Donner la spécification sous-forme de pré et post-conditions, ainsi que les invariants de boucle pour les programmes suivants.

Minimum d'un tableau

```
int min(int t[],int n) {
  int i;
  int m = t[0];

  for(i = 1; i < n; i++) {
    if(t[i] < m) {
      m = t[i];
    }
  }
  return m;
}
```

Tri par sélection

```
void swap(int t[], int i, int j) {
  int tmp = t[i];
  t[i] = t[j];
  t[j] = tmp;
}

void tri_selection(int t[], int n) {
  int i = 0;
  int min;

  while(i < n) {
    int min = i;
    int j = i+1;
    while (j < n) {
      if (t[j] < t[min]) {
        min = j;
      }
      j = j+1;
    }
    swap(t,i,min);
    i = i + 1;
  }
}
```