

## Méthodes formelles de vérification (MF)

### TD n° 4 : Logique de Floyd-Hoare

#### Exercice 1 :

Prouvez que les triplets de Hoare suivants sont valides, ou trouvez un contre-exemple s'ils ne le sont pas et donnez une spécification correcte.

1.  $\{x > 0\} \ x := x+1 \ \{x > 0\}$
2.  $\{\text{true}\} \ x := x+1 \ \{x > 0\}$
3.  $\{x < 0\} \ x := x-1 \ \{\text{true}\}$
4.  $\{x < 0 \vee x = 2\} \ \text{if } (x \geq 0) \ \text{then } x := x+1 \ \text{else skip} \ \{x = 3\}$
5.  $\{x = v_0 \wedge y = v_1\} \ z := x; \ x := y; \ y := z \ \{x = v_1 \wedge y = v_0\}$
6.  $\{\text{true}\} \ \text{while true do skip} \ \{\text{true}\}$
7.  $\{\text{true}\} \ \text{while true do skip} \ \{\text{false}\}$
8.  $\{x = v_0 \wedge v_0 > 0 \wedge y = v_1\}$   
 $\text{while } (x > 0) \ \text{do } y := y-1; \ x := x-1 \ \text{od}$   
 $\{x = 0 \wedge y = v_1 - v_0\}$

#### Exercice 2 :

On considère deux règles de Hoare pour l'affectation :

FORWARD

$$\frac{}{\{P\} \ x := e \ \{P[x/e]\}}$$

BACKWARDS

$$\frac{}{\{P[e/x]\} \ x := e \ \{P\}}$$

Montrer soit que les règles données sont équivalentes ou donner un contre-exemple si elles ne le sont pas. Quelle règle est préférable ?

#### Exercice 3 :

Donner un programme P que valide la spécification suivante :

$$\{x > 0 \wedge z > 0\} \ P \ \{\text{false}\}$$

**Exercice 4 :**

Donner une spécification et sa preuve pour les programmes suivantes :

1. On suppose que  $x > 0$

```
while (x != 0) do
  y := y-1;
  x := x-1
od
```

2. On suppose que  $x > 0$

```
y := 0;
z := 0;
while (y != x) do
  z := z+x;
  y := y+1;
od
```

3. On suppose que  $n > 0$  et  $m > 0$

```
x := m;
y := 0;
while (n <= x) do
  x := x-n;
  y := y+1;
od
```

4. On suppose qu'une liste  $ls$  est donnée

```
i := 1;
lr := [];
while (i <= length(ls)) do
  lr = ls[i]::lr;
od
```

5. Prouver le programme suivante :

```
{ true }
i := 0;
while (i < |a|) {
  a[i] := 0;
}
{  $\sum_{i=0}^{|a|-1} a[i] = 0$  }
```

6. Prouver le programme suivante :

$$\left\{ \sum_{i=0}^{|b|-1} b[i] = 0 \wedge \sum_{i=0}^{|c|-1} c[i] = 0 \right\}$$

```
i := 0;
j1 := 0;
j2 := 0;
while (i < |a|) {
  if (a[i] > 7) {
    b[j1] := a[i];
    j1 := j1 + 1;
  } else {
    c[j2] := a[i];
    j2 := j2 + 1;
  }
  i := i + 1;
}
```

$$\left\{ \sum_{i=0}^{|b|-1} b[i] < |b| * 7 \wedge \sum_{i=0}^{|c|-1} c[i] = |c| * 7 \right\}$$