Méthodes formelles de vérification (MFVerif)

Logique de Floyd-Hoare

Exercice 1 – *Triplets de Hoare*

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

Exercice 2 – *Affectation*

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 où donner un contre-exemple si elles ne le sont pas. Quelle règle est préférable ?

Exercice 3 – *Synthèse de programme*

Donner un programme `P` que valide la spécification suivante :

$$\{x > 0 \land z > 0\}\ \texttt{P}\ \{\texttt{false}\}$$

Exercice 4 – *Preuve en logique de Hoare*

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 que la liste `ls` est donnée

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

5. Prouver le programme suivant :

$\left\{ \ true \ \right\}$

```
i:= 0;
while (i < |a|) {
  a[i]:= 0;
  i:= i + 1
}
```

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

6. Prouver le programme suivant :

$\left\{ \ (\forall i \in [0, |a|], 0 \leqslant a[i]) \ \wedge \ b = [\,] \ \wedge \ c = [\,] \right\}$

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

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