Preuve de programmes

Introduction

- On prouve la correction d'un programme par rapport à sa spécification.
- On manipule deux langages : le langage de programmation et le langage logique de spécification.
  - Langage de programmation : assignations, opérateurs arithmétiques, composition, while, if-then-else, tableaux.
  - Langage de spécification : logique du premier ordre avec égalité et arithmétique.
- On fait une vérification statique en raisonnant sur le programme et sur des assertions logiques à propos de ce programme.

Exemple introductif

```
S:= 0;
N:=1;
while 101 > N do
S:= S+N;
N:=N+1;
done
```

But : démontrer que la valeur finale de `S` est 5050.


```
S:= 0;
N:=1;
```
$\{S = 0 \wedge N = 1\}$
```
while 101 > N do
```
$\{101 \geqslant N\}$
```
S:= S+N;
N:=N+1;
done
```
$\{101 \leq N\}$

Déroulement de l'algorithme

S	N
0	1
$0+1$	2
$0+1+2$	3
$0+1+2+3$	4
$\vdots$	
$0+1+\ldots+k-1$	k

$$S = 1 + 2 + \ldots + (N-1) = \frac{N*(N-1)}{2}$$


```
S:= 0;
N:=1;
```

$\{S = 0 \wedge N = 1\}$

```
while 101 > N do
```

invariant : $\{S = \frac{N*(N-1)}{2}, 101 \geq N\}$

```
   S:= S+N;
   N:=N+1;
done
```

$\{101 \leq N\} \wedge 101 \geq N \wedge S = \frac{N*(N-1)}{2}\} \Rightarrow S = \frac{101*100}{2} = 5050$

Logique de Hoare

Syntaxe : on manipule de formules de la forme

$$\{P\}\ Prog\ \{Q\}$$

où P et Q sont deux formules du calcul des prédicats.

Intuition : si P est une formule vraie avant l'exécution de $Prog$ et si $Prog$ est exécuté et termine, alors la formule Q est vraie après l'exécution de $Prog$. $Prog$ est ainsi un transformateur de prédicats.

On se donne des axiomes et des règles d'inférence pour établir des théorèmes de la forme $\{P\}\ Prog\ \{Q\}$.

Cas spéciaux

$$\{true\}\ x := 0\ \{x = 0\}$$

$$\{false\}\ x := 0\ \{x = 1\}$$

$$\{false\}\ Prog\ \{Q\}$$

Règles de la logique de Hoare

Axiome de substitution :

$$\{Q\{v/exp\}\}\ v := exp\ \{Q\}$$

Example :

$$\begin{array}{lll} \{x = 5\} & x := x + 1 & \{x = 6\} \\ \{x + y = 10\} & z := x + y & \{z = 10\} \\ \{x \geq y\} & x := x - y & \{x \geq 0\} \\ \{z > y + 50\} & x := z - 43 & \{x > y + 7\} \end{array}$$

Règle de composition séquentielle :

$$\frac{\{P\}\ Prog1\ \{R\} \quad \{R\}\ Prog2\ \{Q\}}{\{P\}\ Prog1; Prog2\ \{Q\}}$$

Example :

$$\begin{array}{lll} \{x = b \wedge y = a\} & z := x; x := y; y := z & \{x = a \wedge y = \\ \{x + y = 22\} & x := x + 1; z := x + 2; y := y + z & \{y = 25\} \end{array}$$

Règle du conditionnelle :

$$\frac{\{P \wedge C\}\ Prog1\ \{Q\} \quad \{P \wedge \neg C\}\ Prog2\ \{Q\}}{\{P\}\ \textbf{if}\ \overline{C}\ \textbf{then}\ Prog1\ \textbf{else}\ Prog2\ \{Q\}}$$

$$\frac{\{P \wedge C\}\ Prog\ \{Q\} \quad (P \wedge \neg C) \rightarrow Q}{\{P\}\ \textbf{if}\ \overline{C}\ \textbf{then}\ Prog\ \{Q\}}$$

Nous écrivons « $\overline{C}$ » pour spécifier la condition logique « C » dans la syntaxe du langage de programmation en question.

Example :

$$\{true\}\ \textbf{if}\ a > 0\ \textbf{then}\ b := a\ \textbf{else}\ b := -a\ \{b \geq 0\}$$

Règle de conséquence :

$$\frac{P' \rightarrow P \quad \{P\}\ S\ \{Q\} \quad Q \rightarrow Q'}{\{P'\}\ S\ \{Q'\}}$$

Règle du while :

$$\frac{\{Inv \wedge C\}\ S\ \{Inv\}}{\{Inv\}\ \textbf{while}\ \overline{C}\ \textbf{do}\ S\ \textbf{done}\ \{Inv \wedge \neg C\}}$$

Conseils/remarques :

- L'invariant décrit un état intermédiaire entre l'état initial des variables et l'état final que l'on veut obtenir.
- Écrire les instructions de boucles pour maintenir cet invariant.
- Initialiser les variables pour que l'invariant soit vrai au premier passage.
- Écrire la condition de sortie de boucle en se servant de l'invariant.

Correction et terminaison

Dans ce système on peut prouver un théorème $\{P\}\ Prog\ \{Q\}$ même si $Prog$ ne termine pas ou si il n'est pas exécuté $\Rightarrow$ notion de correction partielle.

Example :

$\{x \leq 10\}\ \textbf{while}\ x \neq 10\ \textbf{do}\ x := x + 1\ \textbf{done}\ \{x = 10\}$

$\{true\}\ \textbf{while}\ x \neq 10\ \textbf{do}\ x := x + 1\ \textbf{done}\ \{x = 10\}$

$\{x > 10\}\ \textbf{while}\ x \neq 10\ \textbf{do}\ x := x + 1\ \textbf{done}\ \{faux\}$

Toutes les règles

$$\frac{}{\{Q\{v/exp\}\}\ v := exp\ \{Q\}} \qquad \frac{\{P\}\ Prog1\ \{R\} \quad \{R\}\ Prog2\ \{Q\}}{\{P\}\ Prog1; Prog2\ \{Q\}}$$

$$\frac{\{P \wedge C\}\ Prog1\ \{Q\} \quad \{P \wedge \neg C\}\ Prog2\ \{Q\}}{\{P\}\ \textbf{if}\ \overline{C}\ \textbf{then}\ Prog1\ \textbf{else}\ Prog2\ \{Q\}}$$

$$\frac{\{P \wedge C\}\ Prog\ \{Q\} \quad (P \wedge \neg C) \rightarrow Q}{\{P\}\ \textbf{if}\ \overline{C}\ \textbf{then}\ Prog\ \{Q\}}$$

$$\frac{\{Inv \wedge C\}\ S\ \{Inv\}}{\{Inv\}\ \textbf{while}\ \overline{C}\ \textbf{do}\ S\ \textbf{done}\ \{Inv \wedge \neg C\}} \qquad \frac{P' \rightarrow P \quad \{P\}\ S\ \{Q\} \quad Q}{\{P'\}\ S\ \{Q'\}}$$

Exercice

Donner des règles pour les opérateurs suivants :

- `do P until C done`
- `let x = P in R`

La pré-condition la plus faible

Si $\{P\}Prog\ \{Q\}$ est un théorème, on dit que P est la pré-condition la plus faible ssi pour tout autre théorème $\{P'\}Prog\ \{Q\}$ on a $P' \rightarrow P$.

On note dans ce cas la propriété P comme $\texttt{wp}(Prog, Q)$. On a donc

$$\{\texttt{wp}(Prog, Q)\}Prog\ \{Q\}$$

Exercice

Compléter tous les détails de la preuve suivante :

$\{true\}$

```
S:= 0;
N:=1;
while 101 > N do
   S:= S+N;
   N:=N+1;
done
```

$\{\texttt{S} = 5050\}$

Exercice

L'algorithme suivant calcule les entiers q et r t.q. $q = x/y$ et $q * y + r = x$

Prouver :

$\{x \geq 0 \wedge y > 0\}$

```
q:=0;
r:=x;
while r >= y do
  r := r - y;
  q := q + 1;
done
```

$\{q * y + r = x \wedge 0 \leq r < y\}$