PREUVES DE PROGRAMMES

Introduction

Une donnée L traitée grâce à un algorithme programmé donne généralement un résultat.
$\forall \lambda \in L$, P définit une séquence de calcul finie qui donne un résultat $P(\lambda) \in R$ ou définit une séquence de calcul infinie.
$F : D \subset L \rightarrow R$
Le programme P est correct si et seulement s'il calcule bien la fonction F.
$\Rightarrow \forall d \in D$, P(d) est défini : il ne boucle pas et calcule bien P(d).

On a deux règles principales :
- *le programme fournit des résultats*
- *le programme fournit les bons résultats*

Comment vérifier ces deux aspects ?
- *la méthode des tests, non exhaustive*
- *la méthode des preuves de programmes ; elle consiste à prouver mathématiquement que P est correct : $\forall d \in D$, P(d) = F(d)*

Exemple : *DIV*

```
début
    r ← a ;                 a, b, q, r sont des variables entières
    q ← 0 ;                 a et b sont initialisées à des valeurs entières A et B
    tant que r ≥ b faire    D : N × N*
        r ← r - b ;         R : N × N
        q ← q + 1 ;         F(a,b) = (q,r) tel que A = bq + r et r < B
    fait
fin
```

La preuve de programme se décompose en deux parties :
- *la preuve de correction partielle*
- *la preuve d'arrêt*

1. Preuve de correction partielle

1.1. But

Si avant l'exécution de DIV, les variables a et b sont respectivement initialisées à $A \in \mathbf{N}$ et $B \in \mathbf{N}^*$, alors après l'exécution de DIV les valeurs q et r vérifient les conditions $A = bq + r$, $r < B$, $r \geq 0$, $q \geq 0$.

$(a = A) \wedge (b = B) \wedge (A \geq 0) \wedge (B \geq 0)$ { DIV } $(A = bq + r) \wedge (q \geq 0) \wedge (r \geq 0) \wedge (r < B)$

On utilise **l'approche axiomatique de Hoare**. Principe :

- *énoncés de la forme P { instructions } Q*
- *si P est vrai avant l'exécution des instructions alors Q sera vrai après*

1.2. Conditions, axiomes et règles de déduction

1.2.1. Les conditions

Ce sont des expressions pouvant prendre les valeurs VRAI et FAUX, des relations entre variables du programme. Elles peuvent être reliées entre elles par des opérations de l'algèbre de Boole ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$).

1.2.2. Les axiomes et les règles

Toute démonstration consiste à déduire un théorème d'un certain nombre d'axiomes en utilisant des règles de déduction.

Exemple : si E { D } S (axiome) et si E'$\rightarrow$ E (relation entre conditions) alors E'{ D } S

1.2.3. Règles permettant de prouver les théorèmes

Règle d'implication

P $\Rightarrow$ Q , Q { instruction } R $\rightarrow$ P { instruction } R (renforcement)
P { instruction } Q , Q $\Rightarrow$ R $\rightarrow$ P { instruction } R (affaiblissement)

Exemples : $A > B \Rightarrow A - B > 0$
$A - B > 0$ { $D = A - B$ } $D > 0$
$\Rightarrow A > B$ { $D = A - B$ } $D > 0$

$X - 1 > 0$ { $X \leftarrow X - 1$ } $X > 0$
$X > 0 \Rightarrow X \neq 0$
$\Rightarrow X - 1 > 0$ { $X \leftarrow X - 1$ } $X \neq 0$

Règle d'affectation

E { X $\leftarrow$ expression } S

Exemples : $X * Y \geq 0$ { $Z \leftarrow X * Y$ } $Z \geq 0$
$W + Q > 12$ { $X \leftarrow W + Q$ } $X > 12$
$(X + Y)^2 = Y$ { $Z \leftarrow X + Y$ } $Z^2 = Y$

Règle de composition

Si E { P } F et F { Q } S alors E { P , Q } S

Exemples : $X - 1 > 0 \{ X \leftarrow X - 1 \} X > 0$
$X > 0 \{ Z \leftarrow X \} Z > 0$
$\Rightarrow X - 1 > 0 \{ X \leftarrow X - 1 ; Z \leftarrow X \} Z > 0$

Règle conditionnelle

Si $E \wedge C \{ P \} S$ *et* $E \wedge \neg C \Rightarrow S$ *alors E { si C alors P } S*

Exemples : vrai { si X > Y alors m ← X sinon m ← Y } m = max (X , Y)
vrai si $X > Y \Rightarrow$ max (X , Y) = X
max (X , Y) = X { m ← X } m = max (X , Y)
$\Rightarrow$ vrai $\wedge$ X > Y { m ← X } m = max (X , Y) (1)
vrai $\wedge$ X > Y $\Rightarrow$ max (X , Y) = Y
max (X , Y) = Y { m ← Y } m = max (X , Y)
$\Rightarrow$ vrai $\wedge$ Y > X { m ← Y } m = max (X , Y) (2)
vrai $\wedge \neg$(X > Y) { m ← Y } m = max (X , Y)
$\Rightarrow$ vrai { si X > Y alors m ← X sinon M ← Y } m = max (X , Y)

Règle du ET

Si E { P } S et E { P } S' alors E { P } S $\wedge$ *S'*

Règle du OU

Si E { P } S ou E'{ P } S alors E $\vee$ *E'{ P } S*

Règle de l'instruction composée

Si E { P } S alors E { début P fin } S

Règle de la boucle tant que

Si $I \wedge B \{ P \} I$ *alors I { tant que B faire P }* $I \wedge \neg B$
où I est un invariant de boucle

2. Preuve d'arrêt

2.1. Idée

Pour prouver qu'un programme d'arrête $\forall$ d $\in$ D, il suffit de prouver que chacune de ses boucles ne peut être exécutée qu'un nombre fini de fois.

2.2. Méthode

Tant que B faire P

Soient X_1, X_2, X_3 … X_n les variables du programme
Soit I une condition invariante pour la boucle et satisfaite avant son exécution
Soit $v_1 \in N_I$, la valeur des variables avant exécution de la boucle
Si $v_1 \in N_{I \wedge \neg B}$, la boucle s'arrête, sinon $v_1 \in N_{I \wedge B}$ et après exécution de la boucle, la valeur des variables du programme est un nouveau vecteur $v_2 \in N_I$
Si $v_2 \in N_{I \wedge \neg B}$, la boucle s'arrête, …

Donc pour prouver l'arrêt de la boucle, il suffit de montrer que la suite ainsi construite v_1, v_2, v_3 … v_n est finie.
$m : N_{I \wedge B} \rightarrow \mathbf{N}$ telle que $I \wedge B \wedge (m(v) = m_0) \{ P \} I \wedge \neg B \wedge (m(v) < m_0) \forall m \in \mathbf{N}$
où m est un ensemble de vecteurs
Si v a une valeur v_i avant l'exécution de P, alors après :

- soit la boucle s'arrête ($\neg B$)
- soit v prend une valeur v_{i+1} telle que $m(v_i) > m(v_{i+1})$

Il en résulte que la suite (v_1, v_2 … v_n) est elle-même finie donc la boucle finit par s'arrêter.

2.3. Exemple

Application à DIV
Invariant $(b = b_0) \wedge (r \geq 0)$
La condition de continuation de la boucle B est $r \geq b$
Pour m, on choisit la fonction $N_{I \wedge B \rightarrow N}$, $v \rightarrow r$ (reste)