Preuve et terminaison d'algorithmes

Lorsqu'on écrit un programme il faut s'assurer qu'il fonctionne correctement dans tous les cas de figures. Bien qu'utile, un jeu de tests ne suffit pas à assurer que le programme se termine et qu'il fonctionne correctement. Dans certaines applications sensibles comme les centrales nucléaires, l'aviation ou les rames de métro automatisées il est vital de pouvoir s'assurer qu'un programme fonctionne tel que prévu.

Réaliser la preuve d'un algorithme c'est :

- Prouver qu'il se termine : c'est la **terminaison**
- Prouver qu'il fait ce qu'on attend de lui : c'est la **correction partielle**

Il y a **correction totale** s'il y a correction partielle et terminaison.

C'est une preuve dans le même sens qu'en mathématiques. Les algorithmes étant, a priori, indépendants du langage utilisé, sont souvent écrits en pseudo-code.

I Terminaison : variant de boucle

En première, la question de la terminaison ne se pose que lorsque l'algorithme contient une ou plusieurs boucles. On ne s'intéressera qu'aux boucles conditionnelles (Tant que) les boucles bornées (Pour) devant a priori s'arrêter.

Remarque :
Il est possible de faire des boucles bornées qui ne s'arrêtent pas :

En Python ce programme s'arrête.

```
n ← 1
Pour i allant de 1 à n :
  n ←n + 1
FinPour
```

Pour montrer qu'un algorithme s'arrête il faut montrer que pour chaque boucle conditionnelle, le test d'entrée dans la boucle sera invalidé au bout d'un temps fini, autrement dit après un nombre fini de passage dans la boucle. Pour cela, on utilise généralement un **variant de boucle.,** c'est une quantité entière positive qui décroît strictement à chaque itération de la boucle. On utilise le fait qu'une suite d'entiers positifs strictement décroissante est nécessairement finie.

Exemple

On considère le programme ci-contre :

Un variant de boucle est 5-i. car c'est un entier qui décroît de 1 à chaque tour de boucle pour s'arrêter à 0.

```
i ← 0
Tant que i < 5 :
  i ←i + 1
FinTantQue
```

II Correction partielle : invariant de boucle

Définition
Un **invariant** est une propriété qui reste vraie tout au long de l'exécution de l'algorithme.

En première, nous chercherons un invariant de boucle car seuls les programmes itératifs sont au programme.
Un **invariant de boucle** est une propriété qui est vérifiée avant l'entrée dans la boucle,
qui est vérifiée à chaque itération de la boucle et qui amène au résultat escompté à la sortie de la boucle.
En pratique, pour prouver de la correction partielle d'un algorithme, après avoir déterminé un invariant, on procède en trois temps :

1. On montre que l'invariant est vérifié avant la boucle (initialisation) ;
2. On montre que si l'invariant est vérifié avant un passage dans la boucle, alors il est préservé après le passage dans la boucle ;
3. On peut conclure sur la valeur finale à la sortie de la boucle.

Exemple 1
La fonction produit doit retourner le produit de a par b ou a est un entier positif et b un nombre quelconque.

```
Fonction produit(a,b)
  m ← 0
  p ← 0
  Tant que m < a :
    m ← m + 1
    p ← p + b
  FinTantQue
  Renvoyer p
FinFonction
```

On souhaite montrer la correction de l'algorithme. Pour cela on veut montrer que $p=m\times b$ est un invariant de boucle.

Démonstration
1. Avant la boucle $m=0$ et $p=0$, donc on a bien $p=m\times b$
2. Posons $m'=m+1$ et $p'=p+b$ les valeurs de m et p en sortie de boucle. .
   Si à l'entrée dans la boucle $p=m\times b$ alors en sortie de boucle :
   $p'=m\times b+b=(m+1)\times b=m'\times b$
   Autrement dit, la propriété est vraie en sortie de boucle lorsqu'elle est vraie en entrée dans la boucle.
3. Puisqu'en sortie de la boucle $m=a, a$ est un entier positif alors on a bien $p=a\times b$

Exemple 2
On donne la fonction ci-contre qui retourne le quotient et le reste de la division euclidienne des entiers positifs a et b.

```
Fonction div_eucli(a,b)
  r ← a
  q ← 0
  Tant que r >= b :
    r ← r - b
    q ← q + 1
  FinTantQue
  Renvoyer q, r
FinFonction
```

On veut montrer que la propriété $a=bq+r$ est un invariant de boucle ce qui montrera la correction partielle de cette fonction.
Démonstration
1. Avant la boucle r = a et q= 0 donc $a=bq+r$.
2. Posons r'= r -b et q'= q + 1 les valeurs de r et q en sortie de boucle.
   Si $a=bq+r$ en entrée de boucle :
   $bq'+r'=b(q+1)+r-b=bq+b+r-b=bq+r=a$
3. r est strictement décroissant en sortie de boucle on a $r<b$ et $a=bq+r$.