Cours Info - 9

Preuve d'algorithme (initiation)

D.Malka

MPSI 2017-2018

Sommaire

Sommaire

Algorithme correct ?

Un algorithme est correct si :

- **il s'arrête,**
- **pour toute entrée, il produit le résultat attendu.**

Algorithme correct ?

On peut :

- **tester quelques entrées. . .**
- **. . . peut prouver qu'il est incorrect mais pas qu'il est correct.**

Il est mieux de prouver logiquement que l'algorithme est correct :

- **terminaison : variant de boucle,**
- **résultat attendu : invariant de boucle.**

Sommaire

Terminaison d'un algorithme

Variant de boucle

Variant de boucle

On appelle *variant de boucle*, une expression :

- **qui est un entier positif tout au long de la boucle,**
- **qui décroît strictement.**

Logique : lorsqu'une suite d'entiers $\{u_i\}$ décroît strictement, il existe un rang N à partir duquel les termes u_i sont négatifs. Or dans la boucle $u_i > 0 \Rightarrow$ l'algorithme termine nécessairement.

Le variant de boucle est souvent le simple contenu d'une variable telle qu'un compteur de boucle.

Terminaison d'un algorithme

Variant de boucle

Un exemple :

Algorithme 1 : Calcul de k^n

Entrées : int n, flottant k

Sorties : flottant p

```
1 Fonction expo(k :float,n :int)
2     c=n
3     p=1
4     tant que c>0 faire
5         p=p*k
6         c=c-1
7     retourner p
```

Terminaison d'un algorithme

Variant de boucle

Montrons que l'algorithme s'arrête.

Dans la boucle c est définie par la suite $\{c_i\}$ telle que, dans la boucle :

$$\begin{cases} c_0 = n \\ c_{i+1} = c_i - 1 \Rightarrow \forall i,\ c_{i+1} < c_i \\ \forall i, c_i > 0 \end{cases}$$

La suite $\{c_i\}$ est entière, positive et strictement décroissante donc l'algorithme s'arrête.

Terminaison d'un algorithme

Généralisation

Pour prouver la terminaison d'un algorithme, on peut utiliser un autre type de variant de boucle telle qu'une suite entière strictement croissante et majorée par exemple. Il faut retenir donc l'esprit de la démonstration et non la lettre.

Terminaison d'un algorithme

Boucle for

Dans le cas d'une boucle for, la terminaison de l'algorithme est trivial si l'itérateur `i` n'est pas modifié par le corps de boucle.

```
1 def factorielle(n):
2     fact=1
3     i=1
4     for i in range(n):
5         fact=fact*i
6     return fact
```

Il est par exemple évident que la fonction `factorielle` ci-dessus termine au bout de n itérations.

Sommaire

Preuve d'un algorithme

Démonstration par récurrence

Démonstration par récurrence

Démonstration par récurrence de la propriété $\mathcal{P}_n$:

- **Initialisation : montrer que $\mathcal{P}_n$ est vraie à partir d'un certain rang n_0.**
- **Hérédité : montrer que $\mathcal{P}_n \Rightarrow \mathcal{P}_{n+1}$.**

Preuve d'un algorithme

Démonstration par récurrence

Exemple

Montrer par récurrence que la suite définie par :

$$\begin{cases} u_0 = 2 \\ u_{n+1} = \frac{1}{3} u_n \end{cases}$$

s'écrit $u_n = \frac{2}{3^n}$.

Preuve d'un algorithme

Invariant de boucle

Invariant de boucle

Pour démontrer que l'algorithme produit l'effet attendu, on utilise un *invariant de boucle* c'est-à-dire une propriété ou une expression qui :

- **est vérifiée avant d'entrer dans la boucle,**
- **reste vraie après chaque itération de la boucle,**
- **dont la valeur au rang n (à la sortie de la boucle), est la preuve de la correction de l'algorithme.**

Démarche similaire au raisonnement par récurrence.

Difficulté : trouver une expression susceptible d'être un invariant de boucle.

Preuve d'un algorithme

Exemple

Un exemple :

Algorithme 2 : Calcul de k^n

Entrées : int n, flottant k

Sorties : flottant p

```
1 Fonction expo(k :float,n :int)
2     c=n
3     p=1
4     tant que c>0 faire
5         p=p*k
6         c=c-1
7     retourner p
```

Preuve d'un algorithme

Exemple

Dans la boucle, deux suites $\{p_i\}$ et $\{c_i\}$ sont définies par récurrence :

$$\begin{cases} p_0 = 1 & \text{et} \quad p_{i+1} = k * p_i \\ c_0 = n & \text{et} \quad c_{i+1} = c_i - 1 \end{cases}$$

On veut montrer qu'en sortie de la boucle $p = k^n$.

Montrons la proposition $\mathcal{P}_i$: $p_i = k^i$ est un invariant de boucle.

Si $\mathcal{P}_i$ est un invariant de boucle alors à la sortie de la boucle $i = n$ et alors $p = p_n = k^n$ (cqfd).

Preuve d'un algorithme

Exemple

Montrons la proposition $\mathcal{P}_i : p_i = k^i$.

Initialisation : pour $i = 0$, $p_0 = 1 = k^0$ et $c_0 = n$ donc $p_0 = k^i$. $\mathcal{P}_0$ est vérifiée.

Récurrence : $\mathcal{P}_i : p_i = k^i$ supposé vraie. Montrons alors que $\mathcal{P}_{i+1} : p_{i+1} = k^{i+1}$ est vraie.

$p_{i+1} = k * p_i = k^i$

$\mathcal{P}_{i+1}$ est vraie.

A la sortie de la boucle $\mathcal{P}_n$ est vraie donc $p_n = k^n$. CQFD ! ! !

Preuve d'un algorithme

Comment trouver l'invariant de boucle ?

Pour déterminer l'invariant de boucle, il faut garder en tête que son évaluation à la sortie (au rang n) de la boucle est l'expression même de la correction de l'algorithme. Autrement dit, il est souvent intéressant de partir de l'expression au rang n de l'invariant de boucle pour en chercher son expression au rang i.

Bref, comme dans tout raisonnement, il faut savoir où va pour déterminer comment y aller.