Fiche TD5 : Terminaison et correction

Exercice 1 *Algorithme mystère*

On considère l'algorithme suivant :

```
Entrée : Un entier naturel n
Sortie :
mystere(n) =
    ploum ← faux
    i ← 1
    Tant que i < n − 1 et ploum = faux
        i ← i + 1
        ploum ← (n mod i = 0)
    Renvoyer ploum
```

1. Donner une spécification la plus précise possible pour cet algorithme.
2. Prouver la terminaison de cet algorithme en exhibant un variant adapté.
3. Prouver la correction de cet algorithme vis à vis de la spécification précisée à la question 1.

Exercice 2 *PGCD*

On considère les deux algorithmes suivants dont la spécification est identique : leur entrée est un couple d'entiers naturels $(a, b) \neq (0, 0)$ et leur sortie est le pgcd de a et b.

```
PGCD_rec(a,b) =
    Si b = 0 alors
        Renvoyer a
    Sinon
        Renvoyer PGCD_rec(b, a mod b)
```

```
PGCD_it(a,b) =
    Tant que b > 0
        r ← a mod b
        a ← b
        b ← r
    Renvoyer a
```

1. Prouver la terminaison de ces deux algorithmes.
2. Prouver la correction de ces deux algorithmes.

Exercice 3 *Evaluation de polynômes*

Dans l'algorithme suivant, la fonction tete renvoie la tête d'une liste, et queue renvoit sa queue.

Entrée : Une liste non vide de réels $[a_0, a_1, ..., a_n]$ représentant le polynôme $P = \sum_{i=0}^{n} a_i X^i$, et un réel a.

Sortie : L'évaluation de P en a, c'est-à dire, le réel $P(a)$.

```
Horner(P, a) =
    Si longueur(P) = 1
        Renvoyer tete(P)
    Sinon
        Renvoyer tete(P) + a × Horner(queue(P), a)
```

1. Montrer que cet algorithme termine.
2. Montrer la correction de cet algorithme.

Exercice 4 *Racines carrées*

On considère les deux algorithmes suivants dont la spécification est identique : leur entrée est un entier n et leur sortie est la racine carrée de n arrondie à l'entier inférieur.

```
racine1(n) =
    r ← 0
    Tant que r × r ≠ n
        r ← r+1
    Renvoyer r
```

```
racine2(n) =
    r ← 0
    s ← 1
    Tant que s ≤ n
        r ← r+1
        s ← s + 2 × r + 1
    Renvoyer r
```

1. Montrer la correction de ces deux algorithmes.
2. Sont-ils totalement corrects par rapport à leur spécification ?

Exercice 5 *Recherche dichotomique*

On considère l'algorithme suivant :

```
Entrée : Un tableau T d'entiers trié dans l'ordre croissant et un entier x.
Sortie : vrai si x ∈ T, faux sinon.
cherche(T, x) =
    debut ← 0
    fin ← longueur(T)
    trouve ← faux
    Tant que (trouve = faux) et (debut < fin)
        milieu ← (debut+fin)/2    #division entière
        Si T[m] = x
            trouve ← vrai
        Sinon
            Si T[m] < x
                debut ← m+1
            Sinon
                fin ← m
    Renvoyer trouve
```

1. Justifier la terminaison de cet algorithme.
2. Exprimer un invariant utile pour la boucle présente dans cet algorithme.
3. L'utiliser pour montrer sa correction.

Exercice 6 *Fonction d'Ackermann*

La fonction d'Ackermann, notée A, est définie récursivement sur $\mathbb{N}^2$ de la façon suivante :

$$A(m,n) = \begin{cases} n+1 \text{ si } m=0 \\ A(m-1,1) \text{ si } m>0 \text{ et } n=0 \\ A(m-1,\, A(m,n-1)) \text{ si } m>0 \text{ et } n>0 \end{cases}$$

1. Calculer $A(4,2)$. Pour information, l'ordre de grandeur du nombre d'atomes dans l'univers est de 10^{80}.
2. Montrer qu'un algorithme qui calcule les valeurs de la fonction d'Ackermann termine.
3. Montrer que pour tout $n \in \mathbb{N}$ on a : $A(1,n) = 2+(n+3)-3$, $A(2,n) = 2\times(n+3)-3$, $A(3,n) = 2^{n+3}-3$ et $A(4,n) = 2^{2^{2^{\cdots^{2}}}}-3$ avec $n+3$ exponentiations empilées.

Culture générale : Ce qu'on appelle aujourd'hui "fonction d'Ackermann" est en fait la reformulation d'une fonction de trois variables due à Ackermann en une fonction de deux variables par la mathématicienne Rózsa Péter. La fonction d'Ackermann est un exemple classique de fonction récursive mais non récursive primitive et a contribué à mieux cerner la notion de calculabilité telle que définie par Turing. La réciproque de la fonction d'Ackermann, notée $\alpha(n)$, intervient dans l'analyse de complexité de certains algorithmes ; elle croît si lentement que pour toute considération pratique on peut faire l'approximation $\alpha(n) \leq 5$.