Aix*Marseille université
Initiative d'excellence
Année 2018-19

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 1 : TD 4

Correction d'algorithmes

1 Correction de structure conditionnelle

On considère l'algorithme suivant :

```
algo1
entrée: x, un entier
sortie: y, un entier différent de 0
début
   si ( x == 0 ) alors  { y = 1 } ;
   sinon  { y = x } ;
fin
```

Montrez que cet algorithme est correct dans le sens où il vérifie la condition sur la sortie.

2 Invariants de boucle

On considère l'algorithme suivant :

```
algo2
entrée: n, un entier positif
sortie: i, un entier égal à n
début
  i = 0;
  tant que (i < n) {
    i = i + 1;
  }
  retourner i;
fin
```

On note n_0 la valeur de n donnée en entrée de l'algorithme.

1. Montrez que l'assertion suivante est un invariant de la boucle :

$$A(i,n) := (0 \leq i \leq n) \wedge (n = n_0).$$

2. Montrez que l'assertion A' ci-dessous n'est pas un invariant de la boucle :

$$A'(i,n) := (0 \leq i < n) \wedge (n = n_0).$$

3. Appliquez le théorème de l'invariant pour prouver la correction partielle de l'algorithme. Vous procéderez de la façon suivante :
   - montrez que si $s_0 = [i \mapsto 0, n \mapsto n_0)$ est un état valide juste avant la boucle, alors $A(s_0)$ est vrai ;
   - le théorème implique alors que pour tout état (i,n) après l'exécution de la boucle, $A(i,n) \wedge (i \geq n)$ est vrai. En déduire que $i = n$.

3 Maximum alambiqué

On considère l'algorithme suivant (déjà étudié dans le TP 3...) :

```
algo3
entrée: n et m, deux entiers positifs
sortie:  i, un entier qui est le max de {n , m}
début
  i = m;
  tant que (i < n) {
    i = i + 1;
  }
 retourner i;
fin
```

1. Vérifiez que l'assertion A suivante est un invariant de la boucle :

$$A := (m \leq i \leq n) \wedge (n = n_0) \wedge (m = m_0)$$

2. L'invariant A est-il vrai au début de la boucle ?
3. Prouvez la correction partielle de l'algorithme en considérant deux cas selon que $m_0 \geq n_0$ ou $m_0 < n_0$.

4 Division par soustractions successives

```
modulo
entrée: a, un entier positif et b, un entier strictement positif
sortie: le modulo de a par b
début
  tant que (a >= b) {
    a = a - b;
  }
  retourner a;
fin
```

Prouvez la correction partielle de l'algorithme `modulo`. Vous utiliserez le fait que si $a \geq b$, alors $a \bmod b = (a - b) \bmod b$, pour obtenir une expression de a en fonction de a_0 et b_0.

5 Factorielle

```
factorielle
entrée: n, un entier positif
sortie: la factorielle de n
début
  i = 0;
  fact = 1;
  tant que (i < n){
    i = i + 1;
    fact = fact * i;
  }
  retourner fact;
fin
```

Prouvez la correction partielle de l'algorithme `factorielle` : vous devez trouver une expression de `fact` en fonction de `i`.

6 Exponentiation

```
expo
entrée : x un entier strictement positif, y un entier naturel
sortie : x^y
début
  i = 0;
  résultat = 1;
  tant que (i != y) {
    résultat = résultat * x;
    i = i + 1;
  }
  retourner résultat;
fin
```

Question 1. Montrer que cet algorithme termine.

Question 2. Montrer que cet algorithme est correct en utilisant un invariant de boucle de la forme

$$A(x, y, i, \textit{résultat}) = \{y = y_0 \wedge x = x_0 \wedge \ldots\}$$

On étudie ensuite un second algorithme (qu'on verra être plus efficace) d'exponentiation, basé sur le codage binaire de la puissance y :

```
expo_rapide
entrée : x un entier strictement positif, y un entier naturel
sortie : x^y
début
  puissance = x;
  résultat = 1;
  tant que (y != 0) {
    si y est impair alors {
      résultat = résultat * puissance;
    }
    puissance = puissance * puissance;
    y = y // 2;
  }
  retourner résultat;
fin
```

Question 3. Montrer la terminaison de cet algorithme.

Question 4. En notant x_0 et y_0 les valeurs initiales de x et de y, montrer que

$$\{x = x_0 \wedge \textit{résultat} \times (\textit{puissance})^y = x^{y_0}\}$$

est un invariant de l'algorithme (en notant que si y est impair, alors y', la nouvelle valeur de y à l'issue de l'itération est égale à $(y-1)/2$ et si y est pair, y' vaut $y/2$).

Question 5. En déduire que l'algorithme est correct.