Outils logiques et algorithmiques – TD 1 – Spécifications et invariants

Exercise 1 (Raisonnements corrects et incorrects) Une mère dit à son fils : « s'il pleut, tu ne sors pas ». Ce que dit la mère est vrai.

1. S'il ne pleut pas, peut-on en déduire que le fils sort ?
2. Le fils sort, peut-on en déduire qu'il ne pleut pas ?
3. Le fils ne sort pas, peut-on en déduire qu'il pleut ?

Bonus, entendu dans un reportage :

4. « Ce pleur de faim est commun à tous les enfants du monde. La preuve avec le petit Paul, 1 mois. »

□

Exercise 2 (Formalisation) On se donne les prédicats suivants qui parlent de personnes et d'aliments :

$\mathsf{aime}(x, y)$	x aime l'aliment y
$\mathsf{mange}(x, y)$	x mange l'aliment y
$x = y$	x et y sont égaux

ainsi qu'une constante moi qui représente la personne qui parle et une constante cdb représentant les choux de bruxelles.

1. Traduire les formules suivantes en langage naturel
   (a) $\forall x, \exists y, \mathsf{mange}(x, y)$
   (b) $\exists y, \forall x, \mathsf{mange}(x, y)$
   (c) $\exists x, \forall y, \mathsf{mange}(x, y)$
   (d) $\forall x\ y\ z, \mathsf{aime}(x, y) \wedge \mathsf{aime}(x, z) \implies (y = z)$
2. Exprimer par des formules logiques sur le langage précédent les propositions suivantes
   (a) J'aime tout ce que je mange
   (b) Je mange tout ce que j'aime
   (c) Je n'aime pas les choux de bruxelles, mais j'en mange
   (d) Si j'aime les choux de bruxelles, alors je mange de tout
   (e) Je ne mange que des choux de bruxelles
3. Les formules 2a et 2b sont-elles équivalentes ?
4. On suppose la formule 2c vraie, peut-on en déduire que 2d est vraie ?

□

Exercise 3 (Carré) L'algorithme suivant calcule le carré d'un nombre entier à l'aide exclusivement d'additions et de multiplications par 2.

```
static int square(int n) {
  int r = 0;
  int i = 0;
  while (i < n) {
    r += 2*i + 1;
    i++;
  }
  return r;
}
```

1. Donner une spécification pour cet algorithme, en précisant à l'aide d'une précondition pour quels entiers il fonctionne effectivement.
2. Donner un invariant pour la boucle, sous la forme d'une relation entre les valeurs de i et r.

□

Exercise 4 (Palindromes) L'algorithme suivant vérifie si une chaîne *s* forme un palindrome, c'est-à-dire si la séquence de caractères lue de gauche à droite est égale à la séquence de caractères lue de droite à gauche. Exemples : `"1001001"` est un palindrome, mais `"Chuck Norris"` n'est pas un palindrome.

```
static boolean palindrome(String s) {
  int i = 0;
  int j = s.length() - 1;
  while (i < j) {
    if (s.charAt(i) != s.charAt(j)) return false;
    i++;
    j--;
  }
  return true;
}
```

1. Donner une spécification de la notion de palindrome.
2. Donner un invariant pour la boucle.

□

Exercise 5 (Minimum) Voici le code pour un algorithme simple de recherche d'un élément minimal dans un tableau.

```
static int minimum(int[] t) {
  assert (t.length > 0);
  int m = t[0];
  for (int i=0; i < t.length; i++) {
    if (t[i] < m)
      m = t[i];
  }
  return m;
}
```

1. Manifestement, le programmeur a tenu à rendre explicite une précondition. Quelle est-elle ?
2. Donner une spécification pour la recherche de minimum.
3. Donner un invariant pour la boucle.
4. Proposer une spécification différente, qui admettrait un tableau arbitraire en entrée (*i.e.* sans précondition).

□

Exercise 6 (Plus longues séquences) La fonction suivante cherche dans une chaîne la longueur maximale d'une séquence répétant un même caractère.

```
static int longestRepetition(String s) {
  int i = 0, r = 0;
  while (i < s.length()) {
    int k = 1;
    while (i+k < s.length() && s.charAt(i+k) == s.charAt(i))
      k++;
    if (k > r)
      r = k;
    i += k;
  }
  return r;
}
```

1. Donner une spécification du problème.
2. Donner des invariants pour les deux boucles.

□