Année 2018-2019 Homework 2

Logique et preuve de programmes
IF107

Filière : Informatique, Année : 1

Sujet de : F. Herbreteau

Un mot est une séquence finie de lettres d'un alphabet Σ. On note $|w|$ la longueur du mot w. Un mot w est vu comme la fonction $w : [0, |w|[\to \Sigma$ qui associe à $w(i)$ la $(i+1)$ème lettre de w. Par exemple $w = aba$ est de longueur $|w| = 3$ et $w(1) = b$. On étudie la terminaison et la correction d'algorithmes permettant de déterminer si un mot w est un palindrôme.

1 Version itérative

On s'intéresse dans un premier temps à un algorithme itératif de spécification suivante :

PRECOND $n \geq 0$

POSTCOND $return \iff \forall k \in [0; n[. w(k) = w(n-k-1)$

Exercice 1 (Version itérative)

Montrer la terminaison et la correction de l'algorithme ci-dessous vis à vis de la spécification ci-dessus.

```
algorithm palindrome_iter(w,n):
  i := 0;
  while (i < n ∧ w(i) = w(n − i − 1)) do
    i := i+1
  endwhile;
  if (i < n) then
    return := false
  else
    return := true
  endif
```

L'algorithme `palindrome_iter` n'indique pas que w est un palindrôme. Qu'indique-t-il exactement ? Comment faut-il l'utiliser pour déterminer si w est un palindrôme ? ⧫

2 Version récursive

On considère maintenant une version récursive de spécification suivante :

PRECOND $i \geq 0 \wedge j \geq 0$

POSTCOND $return \iff \forall k \in [0, j-i]. w(i+k) = w(j-k)$

Exercice 2 (Version récursive)

Montrer la terminaison et la correction de l'algorithme ci-dessous vis à vis de la spécification ci-dessus.

```
algorithm palindrome_rec(w, i, j):
  if (i ≥ j) then
    return true
  else if (w(i) ≠ w(j)) then
    return false
  else
    return palindrome_rec(w, i+1, j−1)
  endif
```

L'algorithme `palindrome_rec` n'indique pas que w est un palindrôme. Qu'indique-t-il *exactement* ? Comment faut-il l'utiliser pour déterminer si w est un palindrôme ? ⧫

3 De l'importance d'écrire simplement les algorithmes

On considère une deuxième version itérative de spécification :

PRECOND $n \geq 0$

POSTCOND $return \iff \forall k \in [0; n[. w(k) = w(n-k-1)$

Exercice 3 (Version itérative 2)

Montrer la la correction de l'algorithme ci-dessous vis à vis de la spécification ci-dessus (la preuve de terminaison est identique au premier exercice).

```
algorithm palindrome_iter2(w,n):
  i := 0; palindrome := true
  while (i < n ∧ palindrome) do
    if (w(i) ≠ w(n−1−i)) then
      palindrome := false
    else
      i := i+1
    end
  endwhile;
  return := palindrome
```

Expliquer *précisément* ce qui rend la preuve de cet algorithme plus compliquée que celle du premier exercice. ⧫