Option Informatique
ESIM 2003 : mots de Lyndon

Corrigé

11 mars 2007

2 Mots de Lyndon

On note $|u|$ la longueur de u (nombre de lettres). Le mot vide est de longueur 0.

Question 1 *On prouve ici que $\leq$ est un ordre total sur les mots. On peut supposer que $n \leq m$ (quitte à échanger les rôles de u et v). Deux cas se présentent :*

- *u est un début de v : on a alors $u < v$*
- *u n'est pas un début de v*

  Par négation on a : $(n > m)$ ou ($n < m$ et il existe au moins un indice $i \leq n$ tel que $u_i \neq v_i$).

  Dans le premier cas : $n = m$, et comme $u \neq v$: il existe au moins un indice $i \leq n$ tel que $u_i \neq v_i$.

  Notons k le plus petit indice convenant, on a donc $1 \leq k \leq n$ et $\forall i < k,\ u_i = v_i$ et $u_k \neq v_k$.

  Si $u_k < v_k$ on a $u < v$. Si $u_k > v_k$ on a $v < u$.

  Ce que l'on voulait

Remarque : comme l'ordre est total, la négation de «$u < v$» est «$u = v$ ou $v < u$».

Question 2 *On suppose qu'il existe un mot v tel que : $u = vw = tv$ avec w et v non vides.*

Raisonnons par l'absurde et supposons que u est un mot de Lyndon. vw et tv ont la même longueur, donc l'un ne peut être début de l'autre. Mais vt est un conjugué de u donc $u = vw < vt$. Avec ce qui précède : $vw << vt$ (1). De même, wv est un conjugué de u donc $u = tv < vw$. Puis $tv < wv$ (2).

w et t ont la même longueur $|u| - |v|$. Notons ncette longueur. On a $n \geq 1$ car w et v sont non vides.

(1) donne $w_1 \leq t_1$ et (2) donne $t_1 \leq w_1$ donc $t_1 = w_1$. Par récurrente $t_i = w_i$ pour $i = 1$ à n.

On en déduit que $w = t$, ce qui est contradictoire avec (1).

Question 3 *Par double implication.*

Supposons que u est un mot de Lyndon.

Soit h une fin de u, c'est à dire que $u = wh$ avec w non vide. Montrons que $u < h$.

Comme hw est conjugué de u, on a $u < hw$. Comme $|u| = |hw|$: u ne peut être un début de hw. Donc $u << hw$. Autrement dit, les lettres de u et de hw conïncident jusqu'à un rang $k - 1$, avec $u_k < (hw)_k$.

Il reste à prouver que $k \leq |h|$. Sinon : h est un début de u puisque toutes ses lettres coïncident avec celles de u et puisque $|h| < |u|$. D'après la question rpécédente, ceci est exclus.

Supposons maintenant que toute fin h de u vérifie $u < h$.

Posons $u = vt$ avec v et t non vides. Montrons que $u < tv$. Par hypothèse, on a déjà : $u < t$. Ceci entraîne $u << t$ comme précédemment. Puis $u << tv$ par définition de $<<$. Puis $u < tv$.

Question 4 *Soient f et g deux mots de Lyndon avec $f < g$. Montrons que fg est un mot de Lyndon. Posons $u = fg$.*

Soit h une fin de u : $u = th$ avec t non vide. Montrons que $u < h$. Ceci terminera la preuve de cette implication (voir la question précédente).

Si $h = g$: on a $t = f$ et $u = fg$. Montrons que $fg < g$. Comme $f < g$ on a : f début de g ou $f << g$.

- *Si $f << g$ on a $fg << g$ par définition de $<<$, et donc $fg < g$.*
- *Si f est un début de g : $g = fw$ avec w non vide. Comme g est un mot de Lyndon et w une fin de g on a $g < w$ puis $g << w$ (à cause des longueurs) puis $fg << fw = g$. On en déduit $fg < g$.*

Si $h \neq g$ on a deux nouveaux cas, à savoir h fin de g ou g fin de h.

- *si h est une fin de g : On a $u = fg < g$ (cas $h = g$) et $g < h$. Donc $u < h$.*
- *si g est une fin de h : On a $h = vg$ où v est une fin de f. Donc $f < v$. C'est à dire $f << v$ (à cause des longueurs). Puis $fg << vg = h$. D'où $u < h$.*

Soit u un mot de Lyndon de longueur > 1. Montrons l'existence de f et g.

Soit g le mot de Lyndon qui est une fin de u et de longueur maximale : $u = fg$ et si $f = vw$ alors wg n'est pas un mot de Lyndon.

Montrons que f est de Lyndon et que $f < g$.

Montrons que $f < g$. On a $f < u$ car f est un début de u. Mais $u < g$ car g est une fin de u et u est de Lyndon. Donc $f < g$ par transitivité.

Montrons que f est de Lyndon. Soit h une fin de f. Montrons que $f < h$, c'est à dire $f << h$ (longueur). On a $f = vh$ et donc $u = vhg$. hg n'est pas de Lyndon : il existe donc une fin t de g telle que $t \leq hg$. On a $f < u$ et $u < t$ donc $f < t$. Puis $f < hg$.

Question 5

```
let inferieur u v = u < v;;
```

Question 6

```
let conjugue u i =
  let n = (string_length u -1) in
  let deb = (sub_string u i (n-i+1)) and fin = (sub_string u 0 i)
  in deb^fin;;
```

Question 7

```
let lyndon u =
  let rep = ref true and n = (string_length u -1) in
  begin
    for i =1 to n do
      if not (inferieur u (conjugue u)) then rep := false;
    done;
   !rep;
  end;;
```

Question 8

```
let rec fusion = function
  [] → []
  | [t] → [t]
  | a : :(b : :q) → if (inferieur a b) then (a^b) : :q else a : :(fusion (b : :q));;

let rec factorisation2 liste =
  let liste_neuve = (fusion liste) in
  if liste_neuve = liste then liste else (factorisation2 liste_neuve);;
```

```
let rec chaineversliste chaine =
  if chaine = "" then []
  else
    let premier = s.[0] and suite = (sub_string chaine 1 (string_length chaine -1)) in
    (char_for_read premier) : :(chaineversliste suite);;

let factorisation chaine = factorisation2 (chaineversliste chaine);;
```

Question 9

$$(0,0,0,0,1),(0,0,0,1,1),(0,0,1,1,1),(0,1,0,1,1),(0,0,1,0,1),(0,1,1,1,1)$$

Question 10

```
let rec insere_mot_lst mot = function
    [] → [mot]
  | t : :q when t=mot → (t : :q)
  | t : :q when (inferieur mot t) → mot : :(t : :q)
  | t : :q → t : :(insere_mot_lst mot q);;
```

Question 11

```
let rec insere_lst_lst lst1 lst2 =
  match lst1 with
      [] → lst2
    | mot : :q → insere_lst_lst q (insere_mot_lst mot lst2);;
```

Question 12

```
let rec fusionne_mot_lst mot = function
    [] → []
  | t : :q when (inferieur mot t) → (mot^t) : :(fusionne_mot_lst mot q)
  | t : :q → (fusionne_mot_lst mot q);;

let rec fusionne_listes lst1 lst2 =
  match lst1 with
      [] → [];
    | mot : :q → insere_lst_lst (fusionne_listes q lst2) (fusionne_mot_lst mot lst2);;
```

Question 13

```
let nmax=20;;
let lynd = make_vect nmax [];;

let remplir_lynd n =
  begin
    lynd.(1) < - ["0";"1"];
    for i = 2 to n do
      for k = 1 to (i-1) do
        let liste = fusionne_listes lynd.(k) lynd.(i-k) in
        lynd.(i) < - insere_lst_lst lynd.(i) liste
      done
    done
  end;;
```