Option Informatique en Spé MP et MP*

Devoir surveillé du mardi 9 février 1999

Sous-mots, mélange de mots, le théorème de Higman

Résumé

À côté de la notion classique de *facteur* d'un mot (ce que les anglo-saxons appellent *subword*), existe aussi la notion de *sous-mot* (que les mêmes baptisent *subsequence* ou *scattered subword*). Un sous-mot u d'un mot v est obtenu en effaçant de v zéro, une ou plusieurs lettres, mais en laissant les autres dans leur ordre initial.

Dans une première partie, on étudie quelques propriétés de la relation d'ordre sur X^* associée à cette notion de sous-mot. C'est aussi l'occasion de rencontrer quelques langages rationnels.

La deuxième partie propose de calculer le nombre de façons dont un mot u peut être obtenu comme sous-mot d'un mot v ; on présente un algorithme de calcul de ce nombre par programmation dynamique.

Dans une troisième partie, on définit le *mélange* (en anglais : *shuffle*) de deux mots ; cette notion est fortement reliée à celle de sous-mot.

Enfin la dernière partie établit un résultat classique dû à Higman : dans un ensemble infini de mots, il existe nécessairement au moins un couple (u, v) de mots distincts tels que u soit sous-mot de v.

Veuillez rédiger chaque partie sur une copie séparée.

Table des matières

Notations, définitions, et mises en garde

▶ Dans tout le problème, X désigne un alphabet ; son cardinal est noté $|X|$. Si u est un mot de longueur p et $0 \leqslant i \leqslant p$, on note $u[1..i]$ le préfixe de longueur i de u. En particulier, $u[1..0] = \varepsilon$.

On peut admettre un résultat, à condition de le signaler clairement ; en tout état de cause, il est demandé de rédiger les questions dans l'ordre de l'énoncé. Les programmes devront être concis, et suffisamment documentés pour être compréhensibles. L'emploi de références est interdit. Les questions marquées ⋆⋆⋆ sont, à mon sens, plus délicates.

▶ On pourra utiliser les fonctions suivantes :

- `mem : 'a -> 'a list -> bool`
  Spécification : `mem x y` est vrai ssi x est membre de la liste y ;
- `union : 'a list -> 'a list -> 'a list`
  Spécification : `union x y` construit la liste des éléments apparaissant dans l'une au moins des deux listes x et y ;
- `map : ('a -> 'b) -> 'a list -> 'b list`
  Spécification : `map f y` construit la liste des images par f des membres de la liste y ;
- `it_list : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a`
  Spécification : `it_list f x y` calcule $f(\ldots(f(f(x, y_1), y_2), \ldots, y_n)$.

1 Sous-mots : définition, quelques propriétés

▶ Les deux premières questions sont indépendantes de la suite de cette partie.

Question 1 • Rédigez en Caml une fonction :

```
list_of_string : string -> char list
```

spécifiée comme suit : `list_of_string s` convertit la chaîne de caractères s en une liste de caractères. Par exemple, avec l'appel :

```
list_of_string "R2D2"
```

on obtiendra la liste `['R';'2';'D';'2']`.

Question 2 • Rédigez en Caml une fonction :

```
string_of_list : char list -> string
```

spécifiée comme suit : `string_of_list l` convertit la liste de caractères ℓ en une chaîne. Par exemple, avec l'appel :

```
string_of_list ['C';'6';'P';'0']
```

on obtiendra la chaîne de caractères `"C6P0"`.

▶ Un mot u de longueur $p \geqslant 1$ est un *sous-mot* d'un autre mot v de longueur $n \geqslant p$ s'il existe une application strictement croissante $s : [\![1, p]\!] \mapsto [\![1, n]\!]$ telle que $u_j = v_{s(j)}$ pour tout $j \in [\![1, p]\!]$. Ceci revient à dire qu'en supprimant certaines lettres du mot v, on obtient le mot u ; par exemple, $u = abb$ est un sous-mot de $v = b\mathbf{a}a\mathbf{b}a\mathbf{b}ba$ (on a **graissé** les lettres de v qui étaient conservées pour former u).

▶ On convient que ε est sous-mot de tout mot u.

▶ On notera $u \prec v$ si u est sous-mot de v. $\mathrm{SM}(v)$ désigne l'ensemble des sous-mots de v : $\mathrm{SM}(v) = \{u \in X^* \mid u \prec v\}$. $\mathrm{SM}(L)$ désigne l'ensemble des sous-mots des mots du langage L : $\mathrm{SM}(L) = \bigcup_{v \in L} \mathrm{SM}(v)$.

Question 3 • Montrez que $u = abb$ est un sous-mot de $v = ababb$. On explicitera toutes les applications s qui conviennent.

Question 4 • Soit u un mot. Montrez que $\mathrm{SM}(u)$ est un langage rationnel.

Question 5 • Rédigez en Caml une fonction :

```
est_sous_mot : string -> string -> bool
```

spécifiée comme suit : `est_sous_mot u v` indique si le mot u représenté par la chaîne de caractères `u` est un sous-mot du mot v représenté par la chaîne de caractères `v`.

Question 6 • Montrez que $\prec$ est une relation d'ordre ; dans quel(s) cas est-ce un ordre total ?

Question 7 • Soit u un mot. On note L_u l'ensemble des mots dont u est un sous-mot : $L_u = \{v \in X^* \mid u \prec v\}$. Montrez que le langage L_u est rationnel.

Question 8 • Dans cette question, $X = \{a, b\}$. Construisez un automate fini déterministe complet qui reconnaît le langage L_{abb}.

Question 9 • Soit L un langage rationnel. Montrez que l'ensemble $\widehat{L}$ des mots v qui ont au moins un sous-mot dans L est un langage rationnel.

Question 10 • Soit L un langage rationnel ; montrez que $\mathrm{SM}(L)$ est lui aussi un langage rationnel.

Question 11 • Exhibez un langage L non rationnel, tel que $\mathrm{SM}(L)$ soit rationnel.

2 Ordre de multiplicité d'un sous-mot

▶ Soient u et v deux mots, de longueurs respectives p et n. On note $\binom{v}{u}$ l'*ordre de multiplicité* de u en tant que sous-mot de v : c'est le nombre d'applications $s : [\![1, p]\!] \mapsto [\![1, n]\!]$ strictement croissantes et telles que $u_j = v_{s(j)}$ pour tout indice $j \in [\![1, p]\!]$; c'est donc le nombre de façons dont u est sous-mot de v. Par convention, $\binom{v}{\varepsilon} = 1$ pour tout mot v, et $\binom{\varepsilon}{u} = 0$ pour tout mot $u \neq \varepsilon$.

Question 12 • Soient v un mot non vide et a une lettre. Que vaut $\binom{v}{a}$?

Question 13 • Soient u et v deux mots, a et b deux lettres. Exprimez $\binom{bv}{au}$ en fonction de $\binom{v}{u}$ et $\binom{v}{au}$, en faisant intervenir le symbole de Kronecker $\delta_{a,b}$ dont on rappelle qu'il est égal à 1 si $a = b$, à 0 sinon.

Question 14 • En appliquant répétitivement la formule que l'on vient d'établir, calculez $\binom{ababb}{abb}$.

Question 15 • Rédigez en Caml une fonction :

```
omul : char list * char list -> int
```

spécifiée comme suit : `omul (u,v)` calcule $\binom{v}{u}$, si le mot u est représenté par la liste de caractères `u` et le mot v par la liste de caractères `v`. On utilisera une formulation récursive, et on ne fera pas appel à un vecteur (ou une matrice) pour stocker des résultats intermédiaires.

Question 16 • Que pensez-vous du coût du calcul avec cette méthode ?

Question 17 • Pour $0 \leqslant i \leqslant n$ et $0 \leqslant j \leqslant p$, on note $m_{i,j} = \binom{v[1..i]}{u[1..j]}$. Expliquez comment organiser le calcul des coefficients de la matrice m pour ramener à un $\mathcal{O}(np)$ le coût du calcul de $\binom{v}{u}$.

Question 18 • Construisez la matrice qui correspond à $u = abb$ et $v = ababb$.

3 Mélange de mots

▶ Soient u et v deux mots. Le *mélange* de ces deux mots, noté $u \circ v$, est l'ensemble des mots qui peuvent s'écrire $u_1v_1u_2v_2 \dots u_nv_n$, où $(u_i)_{1 \leqslant i \leqslant n}$ et $(v_i)_{1 \leqslant i \leqslant n}$ sont deux familles de mots telles que $u = u_1u_2 \dots u_n$ et $v = v_1v_2 \dots v_n$. Par exemple, le mot $\mathbf{a}a\mathbf{b}b\mathbf{a}b\mathbf{b}b$ appartient au mélange des deux mots $u = abb$ et $v = ababb$ (on a **graissé** les lettres qui provenaient du mot u).

▶ Le mélange de deux langages L et M, noté $L \circ M$, est l'ensemble des mots que l'on peut obtenir en mélangeant un mot de L et un mot de M :

$$L \circ M = \bigcup_{u \in L, v \in M} u \circ v$$

$\circ$ est donc une loi de composition sur $\mathcal{P}(X^*)$, et $u \circ v$ n'est finalement qu'un raccourci commode pour désigner $\{u\} \circ \{v\}$. Avec cette interprétation, l'opération de mélange peut être définie par induction structurelle au moyen des règles suivantes :

- (R1) $u \circ \varepsilon = \varepsilon \circ u = \{u\}$ pour $u \in X^*$
- (R2) $au \circ bv = a(u \circ bv) \cup b(au \circ v)$, où $u, v \in X^*$ et $a, b \in X$

Question 19 • Énumérez $ab \circ aab$ (on ne demande pas de preuve détaillée).

Question 20 • Montrez que $\circ$ est commutative.

Question 21 • Soient L, M et N trois langages. Justifiez la relation suivante :

$$L \circ (M \cup N) = (L \circ M) \cup (L \circ N)$$

Question 22 • Montrez que $\circ$ est associative.

Question 23 • Rédigez en Caml une fonction :

```
melange_mots : string -> string -> string list
```

qui calcule le mélange de deux mots. La liste résultat ne devra pas contenir de doublons ; en revanche, l'ordre d'énumération est indifférent.

Question 24 ⋆⋆⋆ • On représente un langage fini par une `string list`. Rédigez en Caml une fonction :

```
melange_langages : string list -> string list -> string list
```

calculant le mélange de deux langages finis. Vous devrez faire en sorte que, si les listes représentant les langages à mélanger sont sans doublons, il en soit de même de la liste résultat ; en revanche, l'ordre d'énumération est indifférent et est donc laissé à votre bon goût.

4 Le théorème de Higman

▶ Soit X un alphabet. Une *antichaîne* sur X est un langage L sur X formé de mots deux à deux incomparables pour la relation $\prec$.

Question 25 • Que peut-on dire d'une antichaîne sur l'alphabet $\{a\}$?

Question 26 • Montrez que, sur l'alphabet $\{a, b\}$, il existe des antichaînes de cardinal arbitrairement grand.

▶ En 1952, Higman a établi à propos de relations d'ordre un résultat qui, dans le cadre de la théorie des langages formels, s'énonce comme suit : sur un alphabet X fini, une antichaîne est nécessairement finie. On se propose de prouver ce dernier résultat.

▶ Nous allons raisonner par l'absurde, en considérant une antichaîne infinie L sur un alphabet X. On note $q = |X|$ et $n = \min_{u \in L} |u|$.

Question 27 • Montrez que $q > 1$.

▶ On peut supposer sans perte de généralité que q est minimal : il n'existe pas d'antichaîne infinie sur un alphabet de cardinal inférieur à q. On peut également supposer (toujours sans perte de généralité) que n est minimal : dans une antichaîne infinie sur un alphabet de cardinal q, les mots ont tous une longueur au moins égale à n.

Question 28 • Montrez que $n > 1$.

Question 29 • Soit u un mot de L de longueur n. On note L' l'ensemble des mots de L dont $u[1..n-1]$ est sous-mot. Montrez que $L \setminus L'$ est nécessairement fini.

▶ Il résulte de la question précédente que L' est infini. Soit $v_1, v_2, \ldots$ une énumération des mots de L'.

Question 30 • Montrez que tout mot v_i de L' peut s'écrire

$$v_i = z_{i,1}u_1z_{i,2}u_2 \ldots z_{i,n-1}u_{n-1}z_{i,n}$$

avec $z_{i,j} \in (X \setminus \{u_j\})^*$ pour tout $j \in [\![1, n-1]\!]$.

Question 31 • Montrez que $z_n \in (X \setminus \{u_n\})^*$.

▶ Pour $1 \leqslant j \leqslant n$, on note $Z_j = \{z_{i,j} \mid i \geqslant 1\}$. Un élément x de Z_j est *maximal* s'il n'existe aucun mot y de Z_j tel que x soit un sous-mot de y, distinct de y.

Question 32 • Montrez que Z_j n'a qu'un nombre fini d'éléments maximaux.

Question 33 ⋆⋆⋆ • En déduire que, ou bien Z_j est fini, ou bien on peut (quitte à procéder à une extraction) supposer $z_{i,j} \prec z_{i+1,j}$ pour tout $i \geqslant 1$.

Question 34 • Concluez alors en mettant en évidence une contradiction.

Question 35 ⋆⋆⋆ • Soit L un langage *quelconque*. Montrez que l'ensemble $\widehat{L}$ des mots v qui ont au moins un sous-mot dans L est un langage rationnel.

Question 36 ⋆⋆⋆ • Soit L un langage *quelconque*. Montrez que $\mathrm{SM}(L)$ est un langage rationnel.

The French school emphasizes the coherence
by using the term "factor" for our "subwords"
and saving the term "subword" for our scattered subwords.
Alexandru Mateescu, Arto Salomaa
Formal Languages : an Introduction and a Synopsis

FIN