Option Informatique en Spé MP et MP*
L-systèmes : le corrigé

Partie 1

Question 1 Récurrence sur k. Nous avons déjà $u_2 = ab = u_0u_1$. Supposons acquise la relation $u_{k+2} = u_k u_{k+1}$. Alors :

$$u_{k+3} = \varphi(u_{k+2}) = \varphi(u_k u_{k+1}) = \varphi(u_k)\varphi(u_{k+1}) = u_{k+1}u_{k+2}$$

Ce qui établit l'assertion au rang $k+1$.

Question 2 Nous avons $|u_0| = |u_1| = 1$ et $|u_{k+2}| = |u_k u_{k+1}| = |u_|k|u_{k+1}|$ pour tout $k \in \mathbb{N}$. Nous raconnaissons la suite de Fibonacci ; avec la convention $F_0 = 0$ et $F_1 = 1$, nous aurons $|u_k| = F_{k+1}$.

Question 3 $u_1 = \varphi(u_0) = \varphi(a) = abc^2$; $u_2 = \varphi(u_1) = \varphi(abc^2) = \varphi(a)\varphi(b)\varphi(c)^2 = abc^2bc^2c^2 = abc^2bc^4$; $u_3 = \varphi(u_2) = \varphi(abc^2bc^4) = \varphi(a)\varphi(b)\varphi(c)^2\varphi(b)\varphi(c)^4 = abc^2bc^4bc^6$; et de même $u_4 = abc^2bc^4bc^6bc^8$.

Question 4 Nous remarquons que $|u_k| = (k+1)^2$ pour $k \in [\![0,4]\!]$. Montrons que cette égalité vaut pour tout $k \in \mathbb{N}$.

Méthode 1 : $|u_{k+1}|_a = |u_k|_a = 1$; $|u_{k+1}|_b = |u_k|_b + |u_k|_a = |u_k|_b + 1$; mais $|u_0|_b = 0$, donc $|u_k|_b = k$. Enfin, $|u_{k+1}|_c = 2|u_k|_b + |u_k|_c + 2|u_k|_a = |u_k|_c + 2k + 2$; avec un télescopage et un changement d'indice, nous obtenons : $|u_k|_c = |u_0|_c + \sum\limits_{0\leqslant j<k} \big(|u_{j+1}|_c - |u_j|_c\big) = \sum\limits_{0\leqslant j<k} 2(j+1) = 2\sum\limits_{1\leqslant j\leqslant k} j = k(k+1)$. Rassemblons ces résultats : $|u_k| = k(k+1) + k + 1 = k^2 + 2k + 1 = (k+1)^2$.

Méthode 2 : avec une récurrence immédiate, $u_k = abc^2bc^4bc^6\ldots bc^{2k}$. Donc $|u_k| = 1 + (3+5+7+\cdots+2k+1) = (k+1)^2$.

Méthode 3 (Sanjeevan) : nous avons $u_{k+1} = \varphi^{k+1}(a) = \varphi^k\big(\varphi(a)\big) = \varphi^k(abc^2) = \varphi^k(a)\varphi^k(b)\varphi^k(c)^2 = u_k bc^{2k}c^2 = u_k bc^{2k+2}$, donc $|u_{k+1}| = |u_k| + 2k + 3$ et on termine avec un télescopage.

Question 5 Fixons $\Sigma = \{a,b,c,d\}$ et définissons φ comme suit : $\varphi(a) = abc^2d^4$, $\varphi(b) = bc^2d^4$, $\varphi(c) = cd^3$ et $\varphi(d) = d$. Je dis que le L-système (Σ, φ, a) convient. Avec les notations précédentes, nous avons en effet :

- $|u_k|_a = 1$;
- $|u_{k+1}|_b = |u_k|_a + |u_k|_b = |u_k|_b + 1$; mais $|u_0|_b = 0$, donc $|u_k|_b = k$;
- $|u_{k+1}|_c = 2|u_k|_a + 2|u_k|_b + |u_k]_c = |u_k]_c + 2(k+1)$; comme $|u_0|_c = 0$, il vient $|u_k|_c = \sum\limits_{0\leqslant j<k} 2(j+1) = 2\sum\limits_{1\leqslant j\leqslant k} j = k(k+1)$;
- $|u_{k+1}|_d = 4|u_k|_a + 4|u_k|_b + 3|u_k|_c + |u_k|_d = |u_k|_d + 4 + 4k + 3k(k+1) = |u_k|_d + 3k^2 + 7k + 4$. Terminons ce calcul avec un télescopage : $|u_k|_d = |u_0|_d + \sum\limits_{0\leqslant j<k}(3j^2 + 7j + 4) = 3\sum\limits_{0\leqslant j<k} k^2 + 7\sum\limits_{0\leqslant j<k} k + 4k = 3\dfrac{k(k-1)(2k-1)}{6} + 7\dfrac{k(k-1)}{2} + 4k = k^3 + 2k^2 + k = k(k+1)^2$.
- Nous pouvons maintenant conclure : $|u_k| = |u_k|_a + |u_k|_b + |u_k|_c + |u_k|_d = 1 + k + k(k+1) + k(k+1)^2 = (k+1)^3$.

Bien entendu, il fallait lire $(k+1)^3$ au lieu de k^3 dans l'énoncé.

Partie 2

Question 6 Méthode 1 : soient u et v appartenant à $R_n(\mathcal{S})$. Notons $p = |u|$, $q = |v|$, $u' = \varphi^n(u)$, $v' = \varphi^n(v)$, $p' = |u'|$, $q' = |v'|$, s l'injection croissante de $[\![1,p]\!]$ dans $[\![1,p']\!]$ telle que $u = u'_{s(1)}\ldots u'_{s(p)}$ et t l'injection croissante de $[\![1,q]\!]$ dans $[\![1,q']\!]$ telle que $v = v'_{t(1)}\ldots v'_{t(q)}$. Définissant alors $\sigma : [\![1,p+q]\!] \to [\![1,p'+q']\!]$ par $\sigma(i) = s(i)$ si $1 \leqslant i \leqslant p$ et $\sigma(i) = p' + t(i-p)$ si $p+1 \leqslant i \leqslant p+q$. σ est clairement une injection croissante, qui vérifie $(uv)_i = (u'v')_{\sigma(i)}$ pour tout $i \in [\![1,p+q]\!]$. Ceci prouve que uv est sous-mot de $u'v' = \varphi^n(u)\varphi^n(v) = \varphi^n(uv)$, donc $uv \in R_n(\mathcal{S})$.

Méthode 2, bien plus rapide : nous avons $u \subseteq \varphi^n(u)$ et $v \subseteq \varphi^n(v)$. Mais la relation $\subseteq$ est compatible avec la concaténation, donc $u\varphi^n(v) \subseteq \varphi^n(u)\varphi^n(v) = \varphi^n(uv)$. Par transitivité, $uv \subseteq \varphi^n(uv)$.

Question 7 Soient u et v tels que $uv \in R_n(\mathcal{S})$. Notons $u' = \varphi^n(u)$, $p' = |u'|$, $v' = \varphi^n(v)$ et $q' = |v'|$: uv est sous-mot de $u'v'$, donc u est sous-mot de $u'v'$; notons j le plus petit indice tel que u soit sous-mot de $u'v'[1..j]$. Si $j \leqslant p'$, alors u est sous-mot de u', et $u \in R_n(\mathcal{S})$. Sinon, v est sous-mot de $uv[j+1..p'+q']$, donc de v' ; dans ce cas, $v \in R_n(\mathcal{S})$.

▶ Remarque : s'il existe un exposant n_0 tel que $\varphi^{n_0}(u) = \varepsilon$, alors $\varphi^n(u) = \varepsilon$ pour tout $n \geqslant n_0$.

Question 8 Soit u un mot récurrent pour le L-schéma (Σ, φ) et k un exposant de récurrence de u. Alors u est sous-mot de $\varphi^{pk}(u)$ pour tout $p \geqslant 1$, donc il existe dans la suite de terme général $\varphi^j(u)$ une infinité de termes distincts de ε. Donc aucun terme de cette suite n'est égal à ε, et u est vivant.

• Supposons u vivant. Aucun des $\varphi^j(u)$ n'est égal à ε. Si une infinité de termes de la suite des $\varphi^j(u)$ sont distincts, alors le lemme de HIGMAN permet d'affirmer qu'il existe des exposants j et k distincts tels que $\varphi^j(u) \sqsubseteq \varphi^k(u)$. Si $j < k$, une récurrence immédiate montre que $\varphi^j(u)$ (qui n'est pas vide) est sous-mot de $\varphi^{j+q(k-j)}(u)$ pour tout $q \in \mathbb{N}$. Si $j > k$, alors, comme le morphisme φ n'est pas effaçant, on a $\left|\varphi^j(u)\right| \geqslant \left|\varphi^k(u)\right|$; mais, comme $\varphi^j(u)$ est sous-mot de $\varphi^k(u)$, on a aussi l'inégalité inverse. Donc $\varphi^j(u)$ et $\varphi^k(u)$ ont même longueur, puis, comme l'un est sous-mot de l'autre, sont égaux. Dans les deux cas, la suite engendrée possède une infinité de termes distincts de ε, ce qui montre que u est vivant.

• Variante, pour le cas $j > k$ (SANJEEVAN) : notons $r = j - k$, alors $\varphi^{k+pr}(u) \subseteq \varphi^k(u)$ pour tout $p \in \mathbb{N}$. La suite des longueurs des mots $\varphi^{k+pr}(u)$ est décroissante, donc stationne à partir d'un certain rang p_0 ; alors $v = \varphi^{k+p_0 r}(u)$ est égal à $w = \varphi^{k+(p_0+1)r}(u)$, est sous-mot de w : ceci prouve que v est récurrent.

Question 9 Soit u un mot récurrent de longueur minimale. Si $|u| = 1$, c'est fini. Sinon, notons $u = xv$ avec $x \in \Sigma$. Alors, d'après le résultat de la question 7, ou bien x est récurrente (contradiction), ou bien v est récurrent (contradiction à nouveau). Donc u est une lettre récurrente.

▶ *La question 10 était identique à la question 9 ...*

Question 11 D'après le résultat de la question précédente, les mots n-récurrents minimaux sont les lettres n-récurrentes.

Question 12 Chaque lettre a est récurrente ; notons ν_a un exposant de récurrence de a. Alors le ppcm des ν_a, où a décrit l'alphabet Σ, est un exposant de récurrence de n'importe quel mot récurrent.

Question 13 Dressons un tableau indiquant les lettres qui apparaissent dans $\varphi^n(x)$ pour toute lettre x, et pour des valeurs croissantes de n :

n	1	2	3	4	5
a	bd	ac	abd	$abcd$	$abcd$
b	c	a	bd	ac	abd
c	a	bd	ac	abd	$abcd$
d	a	bd	ac	abd	$abcd$

La dernière colonne montre que $n = 5$ est un exposant de récurrence commun à toutes les lettres de Σ ; et l'examen des quatre colonnes précédentes montre que c'est le plus petit.

Remarque : avec le résultat de la question 6, n est aussi le plus petit exposant de récurrence commun à tous les mots.

Question 14 Notons α_k l'ensemble des lettres qui apparaissent dans les mots $\varphi^j(a)$, $1 \leqslant j \leqslant k$. S'il existe un $k \in [\![1, |\Sigma| - 1]\!]$ tel que $a \in \alpha_k$, alors a est récurrente, et son exposant de récurrence est le plus petit k tel que $a \in \alpha_k$. Sinon, la suite des $|\alpha_k|$ est croissante, à valeurs dans l'intervalle discret $[\![1, |\Sigma| - 1]\!]$ donc stationne à partir d'un rang j appartenant à cet intervalle ; a n'appartient pas à α_j ; à plus forte raison, a n'appartient à aucun α_k et n'est donc pas récurrente.

• Le schéma $a_i \to a_{i+1}$ pour $1 \leqslant i < n$, $a_n \to a_1$ montre que l'on peut avoir $n_a = |\Sigma|$.

Question 15 Remarquons que, si $x \sqsubseteq y$, alors $\varphi(x) \sqsubseteq \varphi(y)$. Soient donc p et q deux éléments de $\boldsymbol{E}(u)$: alors $u \sqsubseteq \varphi^p(u)$ et $u \sqsubseteq \varphi^q(u)$. Donc $\varphi^p(u) \sqsubseteq \varphi^p\left(\varphi^q(u)\right) = \varphi^{p+q}(u)$. Par transitivité, $u \sqsubseteq \varphi^{p+q}(u)$: ceci montre que $p + q \in \boldsymbol{E}(u)$.

• Le résultat est banal pour $q = 1$; s'il est vrai pour $q \geqslant 1$, alors pq et p sont dans $\boldsymbol{E}(u)$, donc $pq + p = p(q+1)$ est encore dans $\boldsymbol{E}(u)$ d'après le résultat précédent. Par récurrence, $pq \in \boldsymbol{E}(u)$ pour tout $q \geqslant 1$.

Question 16 Soit p un exposant de récurrence commun à u et b : $u \sqsubseteq \varphi^p(u)$ et $v \sqsubseteq \varphi^p(v)$. Alors $uv \sqsubseteq \varphi^p(u)\varphi^p(v) = \varphi^p(uv)$ donc p est un exposant de récurrence de uv ; ceci établit $\boldsymbol{E}(u) \cap \boldsymbol{E}(v) \subset \boldsymbol{E}(uv)$.

• Soit p un exposant de récurrence de uv : alors $uv \sqsubseteq \varphi^p(uv) = \varphi^p(u)\varphi^p(v)$; Donc (cf. question 7) ou bien $u \sqsubseteq \varphi^p(u)$, auquel cas $p \in \boldsymbol{E}(u)$; ou bien $v \sqsubseteq \varphi^p(v)$, auquel cas $p \in \boldsymbol{E}(v)$. Dans tous les cas, $p \in \boldsymbol{E}(u) \cup \boldsymbol{E}(v)$, ce qui établit la deuxième inclusion.

Question 17 L'inclusion de gauche à droite résulte de l'inclusion $\Sigma \subset \Sigma^+$. Pour l'inclusion inverse, il suffit de noter que si n est tel que toute lettre est n-récurrente, alors tout mot est n-récurrent.

Question 18 L'inclusion de droite à gauche est banale. Inversement, soit $n > 0$ tel qu'il existe un mot u (non vide) n-récurrent. Sans perte de généralité, nous pouvons supposer u de longueur minimale. Mais alors, si $|u| \geqslant 2$, nous avons $u = xv$ avec $x \in \Sigma$. v, strictement plus court que u, ne peut être n-récurrent. Or, d'après la deuxième inégalité établie à la question 16, l'un au moins des deux mots x et v est n-récurrent : c'est donc x, et la preuve est achevée.

Partie 3

Question 19 La relation $\sqsubseteq$ peut être vue comme une fonction de $\Sigma^* \times \Sigma^*$ dans $\{\mathbf{VRAI}, \mathbf{FAUX}\}$, qui est parfaitement définie par les propriétés suivantes (a et b sont deux lettres) : $\varepsilon \sqsubseteq y = \mathbf{VRAI}$; $x \sqsubseteq \varepsilon = \mathbf{FAUX}$ si $x \neq \varepsilon$; $ax \sqsubseteq ay = x \sqsubseteq y$; $ax \sqsubseteq by = ax \sqsubseteq y$ si $a \neq b$. La traduction en Caml est immédiate :

```
let est_sous_mot x y =
 let m = string_length x and n = string_length y in
 let rec esm_aux i j =
  i = m or
  (j < n &
  if x.[i] = y.[j] then esm_aux (i+1) (j+1)
                   else esm_aux i (j+1) )
 in esm_aux 0 0;;
```

Question 20 Pour toute loi $*$ associative, le calcul en Caml de $a_1 * a_2 * \ldots * a_n$ se fait avec un `it_list` :

```
let flat_string_list = it_list (prefix ^) "" ;;
```

Question 21 Utilisation adroite des fonctions essentielles de la bibliothèque Caml ; `assoc` exploite la *liste d'association* définissant φ, en évitant de réinventer la roue.

```
let calcule_image phi s =
 flat_string_list(map(fun c -> assoc c phi) (list_of_string s)) ;;
```

`list_of_string`, de signature `string -> char list`, ne fait pas partie de la bibliothèque Caml ; en voici une écriture simple, utilisant l'inxoydable `intervalle` :

```
let list_of_string s =
 let l = intervalle 0 (string_length s - 1) in
 map (fun i -> s.[i]) l ;;
```

Question 22 Nous avons besoin d'une fonction `pos` qui indique la position de la première occurrence d'un caractère c dans une chaîne s ; à défaut, le résultat rendu est -1.

```
let pos c s =
 let n = string_length s in
 let rec aux = function
  | k when k=n -> - 1
  | k when s.[k] = c -> k
  | k -> aux (k+1)
 in aux 0 ;;
```

Il suffit ensuite de calculer les images de c par les $n = |\Sigma|$ premiers itérés de φ.

```
let expo_rec c phi =
 let n = list_length phi in
 let rec itere s = function
  | k when k > n -> None
  | k -> let s' = calcule_image phi s in
     if pos c s' >= 0 then Some k else itere s' (k+1)
 in itere (make_string 1 c) 1 ;;
```

FIN