Corrigé DS3-Mines Ponts

Exercice

1. Par définition, le langage $\{\epsilon\}$ est régulier. Soit $n \in \mathbb{N}$. Supposons que, pour tout mot m de longueur n, $\{m\}$ est un langage régulier. Soit m' un mot de longueur $n+1$. $m' = l.m$ où $l \in \Sigma$ et m est un mot de longueur n. Par hypothèse de récurrence, $\{m\}$ est régulier. Par définition $\{l\}$ est régulier. Par stabilité des langages réguliers par concaténation, le langage $\{m'\} = \{l\}.\{m\}$ est régulier. Par récurrence, pour tout mot m, $\{m\}$ est régulier.

Soit p le cardinal de Σ et $(l_i)_{1\leq i\leq p}$ une liste des éléments de Σ. $\Sigma = \bigcup_{i=1}^{p} \{l_i\}$. Les langages réguliers étant stables par union finie, Σ est un langage régulier. Par stabilité par étoile de Kleene, Σ^* est régulier. Par stabilité des langages réguliers par concaténation, $\Sigma^*.\{f\}.\Sigma^*$ est régulier. L'ensemble des mots ayant f comme facteur est donc régulier.

2. Le langage des mots contenant au moins deux occurrences disjointes du facteur f est $\Sigma^*.\{f\}.\Sigma^*.\{f\}.\Sigma^*$. Par concaténation de langages réguliers, c'est un langage régulier.

3. Un mot contient au moins deux occurrences non disjointes de f si et seulement s'il existe $m_1 \in \Sigma^*$, $m_2 \in \Sigma^*$, $m_3 \in \Sigma^*$, tel que f est un préfixe de m_2, f est un suffixe de m_2, $|m_2| > |f|$ et $m = m_1.m_2.m_3$; ceci équivaut à $m = m_1m_2m_3$, $m_1 \in \Sigma^*$, $m_3 \in \Sigma^*$, $m_2 \in \{f\}.\Sigma^*$, $m_2 \in \Sigma^*.\{f\}$, $m_2 \in \Sigma^{|f|+1}.\Sigma^*$. Le langage de ces mots est donc $\Sigma^*.\big((\{f\}.\Sigma^*) \cap (\Sigma^*.\{f\}) \cap \Sigma^{|f|+1}.\Sigma^*\big).\Sigma^*$

Σ est régulier donc $\Sigma^{|f|+1}$ est régulier par concaténation de langages réguliers. Σ^* est régulier, $\{f\}$ est régulier donc, par concaténation, le langage des mots ayant au moins deux occurrences de f est régulier.

4. Soit L_1 l'ensemble des mots ayant une seule occurrence de f, $L_{\geq 1}$ celui des mots ayant au moins une occurrence de f, $L_{\geq 2}$ celui des mots ayant au moins deux occurrences non disjointes de f. Alors $L_1 = L_{\geq 1} \setminus L_{\geq 2}$. D'après les questions précédentes, $L_{\geq 2}$ et $L_{\geq 1}$ sont réguliers. Comme complémentaire d'un langage régulier, $\Sigma^* \setminus L_{\geq 2}$ est régulier donc $L_1 = L_{\geq 1} \cap (\Sigma^* \setminus L_{\geq 2})$ est régulier par intersection de langages réguliers.

Problème

1. Soient $s = aaa$ et $t = aa$. La longueur de s est 3, celle de t est 2. t est l'occurence 2 et 3 de s. Or $1 < 2$ et $1 \geq 2 - 3 + 1$. La réponse est donc OUI.

2. La première occurence possible est k. La dernière est n. L'ensemble des occurrences est inclus dans $[\![k, n]\!]$. Il y en a donc au plus $n - k + 1$. En outre, si $s = a^n$ et $t = a^k$, pour tout $l \in [\![k, n]\!]$, $s = a^{l-k}ta^{n-l}$. Ainsi, t est l'occurrence $k - l$ est s donc l'ensemble des occurrences de t dans s est $[\![k, n]\!]$. Il y en a donc $n - k + 1$.

3.

```
let rec longueur l = match m with | [] -> 0
                                  | t::q -> 1+ longueur q ;;
```

La complexité est $O(n)$.

4.

```
let rec prefixe s t = match s,t with | [] , _ -> true
                                     | _ , [] -> false
                                     | l::s0 , k::t0 -> l=k && prefixe s0 t0 ;;
```

La fonction renvoie un résultat si l'une des deux listes est vide ; une opération de complexité bornée (une comparaison) et, au plus, un appel récursif est effectué sur deux listes de taille un de moins sinon ; la complexité est donc $O\big(\min(n, k)\big)$.

5.

```
let recherche_naive s t = let k = longueur s in
    let rec aux s t p = match t with
        | [] -> []
        | l::t0 -> if prefixe s t then p::(aux s t0 (p+1)) else aux s t0 (p+1) in aux s t k;;
```

6. Les calculs initiaux de longueur ont une complexité $O(n)$ et $O(k)$. Il y a ensuite n appels récursifs et, au cours de chaque appel récursif, un appel à la fonction préfixe sur des listes de taille k et $n - i$ (pour le i-ème passage de boucle). Au total, la complexité est $O(k) + O(n) + \sum_{i=1}^{n} (\min(k, n - i)) = O\big(n \min(k, n)\big)$.

7. Pour tout $q \in Q$, $\rho(q) < q$. Soit $q \in Q$. $\rho^0(q) = q \leq q - 0$. Soit $j \in \mathbb{N}$. Supposons que $\rho^j(q) \leq \max(0, q - j)$. Si $\rho^j(q) = 0$, $\rho^{j+1}(q) = 0$; sinon, $1 \leq \rho^j(q) \leq q - j$ donc $\rho^{j+1}(q) = \rho(\rho^j(q)) \leq \rho^j(q) - 1 \leq q - j - 1$. Dans tous les cas, $\rho^{j+1}(q) \leq \max(0, q - j - 1)$. Par récurrence, pour tout $j \in \mathbb{N}$, $\rho^j(q) \leq \max(0, q - j)$. En particulier, $\rho^q(q) = 0$. De ce fait, $\delta(\rho^q(q), \alpha)$ est défini.

8. Il suffit de placer une transition de 1 vers 1 étiquetée par a, une transition de 2 vers 0 étiquetée par b, une transition de 3 vers 2 étiquetée par b et une transition de 3 vers 1 étiquetée par a.

9. Il s'agit du langage des mots ayant pour suffixe aba.

10.

```
let copie_afdr a = let fc=Array.copy a.final and rc=Array.copy a.repli and k=Array.length a.transition and
tc=Array.make_matrix k lambda 0 in for i=0 to k-1 do  for j=0 to lambda-1 do tc.(i).(j) <- a.t.(i).(j) done done;
{final=fc ; transition=tc ; repli=rc};;
```

11.

```
let enleve_repli a = let ac=copie_afdr a and k=Array.length a.final in
for j=0 to lambda-1 do for i=1 to k-1 do if ac.transition.(i).(j)=-1 then ac.transition.(i).(j) <- ac.transition.(ac.repli(i)).(j)
done done; ac;;
```

La fonction contient une double boucle de taille λ pour la première, k pour la seconde. Sa complexité est donc $O(\lambda k)$.

12.

etat :=0 , liste :=[]

Pour i allant de 1 à n faire :
etat :=delta(etat,ui)
si etat est final alors ajouter i à liste
Fin pour
Renvoyer liste

13.

```
let occurrences a liste = let rec aux etat mot i = match mot with
    | [] -> []
    | u::mot0 -> let etat0=a.transition.(etat).(u) in
    if a.final.(etat0) then i::(aux etat0 mot0 (i+1)) else aux etat0 mot0 (i+1) in  aux 0 liste 1;;
```

Le nombre d'instructions pour chaque appel récursif est borné. Comme il y a n appels récursifs, la complexité est $O(n)$.

14. Cet automate est $\mathcal{A} = (6, \{5\}, \delta, \rho)$ où $\delta(0,a) = 1$, $\delta(0,b) = \delta(0,c) = 0$, $\rho(0) = 0$; $\delta(1,b) = 2$, $\rho(1) = 0$; $\delta(2,a) = 3$, $\rho(2) = 0$; $\delta(3,b) = 4$, $\rho(3) = 1$; $\delta(4,c) = 5$, $\rho(4) = 2$; $\rho(5) = 0$.

15. Il s'agit du langage des mots ayant pour suffixe s.

16. Supposons que $u_1 \cdots u_k$ est un suffixe strict de $u_1 \cdots u_i$. Il existe m un mot non vide tel que $u_1 \cdots u_i = mu_1 \cdots u_k$. De ce fait, $u_k = u_i$ et $u_1 \cdots u_{j-1}$ est un suffixe de $u_1 \cdots u_{i-1}$. De plus, $u_1 \cdots u_{\rho(i-1)}$ est le plus grand suffixe strict de $u_1 \cdots u_{i-1}$ du type $u_1 \cdots u_k$ donc $u_1 \cdots u_{k-1}$ est un suffixe de $u_1 \cdots u_{\rho(i-1)}$. S'il existe $j \in \mathbb{N}$ tel que $u_1 \cdots u_{\rho^j(i-1)}$ est un suffixe strict de $u_1 \cdots u_{k-1}$, par définition de $\rho^{j+1}(i-1) = \rho\big(\rho^j(i-1)\big)$, $u_1 \cdots u_{\rho^{j+1}(i-1)}$ est un suffixe de $u_1 \cdots u_{\rho^j(i-1)}$ donc un suffixe strict de $u_1 \cdots u_{k-1}$.
Or il n'existe pas de suite infinie de suffixes stricts de $u_1 \cdots u_{k-1}$. De ce fait, il existe $j_0 \in \mathbb{N}^*$ tel que $u_1 \cdots u_{k-1} = u_1 \cdot u_{\rho^{j_0}(i-1)}$. Ces mots ayant la même longueur, $\rho^{j_0}(i-1) = k-1$. Ainsi, $\delta\big(\rho^{j_0}(i-1), u_i\big) = \delta\big(k-1, u_k\big) = k$ donc $\delta\big(\rho^{j_0-1}(\rho(i-1)), u_i\big)$ est défini. En particulier, en choisissant $k = \rho(i)$, il existe $j_0 \in \mathbb{N}^*$ tel que $\delta\big(\rho^{j_0}(i-1), u_i\big)$ est défini et vaut $\rho(i)$.
Supposons qu'il existe $j < j_0$ tel que $\delta\big(\rho^j(i-1), u_i\big)$ existe. Posons $k = \rho^j(i-1)$. Alors $\rho^{j_0}(i-1) = \rho^{j_0-j}(k)$ donc $u_1 \cdots u_{\rho^{j_0-j}(k)}$ est un suffixe strict de $u_1 \cdots u_{\rho^j(i-1)}$ donc $u_1 \cdots u_{\rho^{j_0}(i-1)}$ est un suffixe strict de $u_1 \cdots u_k$. Comme $\delta\big(\rho^j(i-1), u_i\big)$ est défini, $u_{\rho^j(i-1)+1} = u_i$ donc $u_{k+1} = u_i$ donc $u_1 \cdots u_{\rho^{j_0}(i-1)} u_i$ est un suffixe strict de $u_1 \cdots u_{k+1}$ donc, comme $u_1 \cdots u_{\rho^{j_0}(i-1)} u_i = u_1 \cdots u_{\rho(i)}$, $u_1 \cdots u_{\rho(i)}$ est un suffixe strict de $u_1 \cdots u_{k+1}$. En outre, $k < i-1$ donc $k+1 < i$ donc $u_1 \cdots u_{k+1}$ est un suffixe strict de $u_1 \cdots u_i$ qui possède comme suffixe strict $u_1 \cdots u_{\rho(i)}$. Ceci contredit la définition de ρ. Ceci prouve qu'il n'existe pas $j < j_0$ tel que $\delta\big(\rho^j(i-1), u_i\big)$ existe.
Finalement, $\rho(i) = \delta\big(\rho^{j_0-1}(\rho(i-1)), u_i\big)$ est défini et $j_0 - 1$ est le plus petit entier naturel vérifiant que $\delta\big(\rho^j(\rho(i-1)), u_i\big)$ est défini.

17.

```
let automate_kmp s = let k=longueur s in let f=Array.make (k+1) false in f.(k) <- true;
  let t = Array.make_matrix (k+1) lambda -1 and r=Array.make (k+1) 0 in for j=0 to lambda -1 do t.(0).(j) <- 0 done;
  let rec aux mot i j = match mot with
     | [] -> ()
     | u::mot0 -> if t.(j).(u)>-1 then begin t.(i).(u) <- i+1; r.(i) <- t.(j).(u); aux mot0 (i+1) r.(i) end else aux mot i r.(j)
  in aux s 0 0; {final = f ; transition = t ; repli = r};;
```

18. Pour tout $k \in [\![0, j_i]\!]$, $\rho^k\big(\rho(i-1)\big) \leq \rho(i-1) - k$. En effet, $\rho^0\big(\rho(i-1)\big) = \rho(i-1)$. Supposons que $k \in [\![0, j_i - 1]\!]$. Alors $\delta(\rho^k(\rho(i-1)), u_i)$ n'est pas défini donc $1 \leq \rho^k\big(\rho(i-1)\big)$. Supposons que $\rho^k\big(\rho(i-1)\big) \leq i-1-k$. Par stricte décroissance de ρ, $\rho^{k+1}\big(\rho(i-1)\big) \leq \rho(i-1) - k - 1 = \rho(i-1) - (k+1)$. Par récurrence, pour tout $k \in [\![0, j_i]\!]$, $\rho^k\big(\rho(i-1)\big) \leq \rho(i-1) - k$. En particulier, $\rho^{j_i}\big(\rho(i-1)\big) \leq \rho(i-1) - j_i$. Enfin, par définition de δ, pour tout état j, si $\delta(j, u_i) = j+1$ donc $\delta(\rho^{j_i}\big(\rho(i-1)\big), u_i) \leq \rho(i-1) - j_i + 1$ donc $\rho(i) \leq \rho(i-1) - j_i + 1$.
Par conséquent, $\sum_{i=1}^{k} j_i \leq \sum_{i=1}^{k} \big(\rho(i-1) - \rho(i) + 1\big) = 0 - \rho(k) + k \leq k$ donc $\sum_{i=1}^{k} j_i = O(k)$.

19.
La création du tableau des états finals est en $O(k)$, sa modification en $O(1)$; la création du tableau des transitions est en $O(\lambda k)$, sa modification en $O(k)$; la création du tableau de repli est en $O(k)$, sa modification en $O\big(\sum_{i=1}^{k} j_i\big) = O(k)$. La complexité totale est $O(k\lambda)$.

20.

```
let recherche_kmp s t = occurrences (enleve_repli (automate_kmp t)) s;;
```

La fonction "automate_kmp" a une complexité de $O(\lambda k)$, la fonction "enleve_repli" a une complexité $O(\lambda k)$; la fonction "occurrences" a une complexité $O(n)$. La complexité totale est $O(n) + O(k\lambda) + O(k\lambda) = O(n + \lambda k)$.
Si $n \leq \lambda$, cette complexité est meilleure que celle de la fonction 6. Cette hypothèse est très fréquente (un texte étudié a souvent une taille supérieure à celle de l'alphabet étudié).

21. Il s'agit du langage des mots qui ont pour suffixe aa, ab ou ba.

22. On considère l'AFDR dont le nombre d'états est 6. Les états finals sont $\{3, 4, 5\}$. Les transitions sont $\delta(0,b) = 1$, $\delta(1,a) = 2$, $\delta(1,c) = 3$, $\delta(2,a) = 4$, $\delta(2,b) = 5$. La fonction de replis est $\rho(0) = 0$, $\rho(1) = 0$, $\rho(2) = 0$, $\rho(3) = 0$, $\rho(4) = 0$, $\rho(5) = 1$.

23.

```
let rec recherche_dictionnaire s t  = match s with | [] -> []
                                                   | s0 :: q -> (recherche_kmp s0 t)::(recherche_dictionnaire q t);;
```

La complexité est $|S|$ fois la complexité de "recherche_kmp" ; c'est donc $O\big(|S| \times (n + k\lambda)\big)$.

24. On considère un état initial 0. Si les premières lettres des motifs sont $\{u_1, \cdots, u_p\}$, on place une transition de 0 vers un état e_{u_1} étiquetés par u_i. Pour tout sous-motif $u_1 \cdots u_j$ de l'un des motifs de S, si $\delta(0, u_1 \cdots u_{j-1}) = e_i$, on définit une transition de e_i vers un état e'_i étiqueté par u_j. Pour tout état e_i auquel on aboutit par un motif $u_1 \cdots u_j$, on considère le plus grand $j' < j$ tel que $u_1 \cdots u_{j'}$ est un suffixe de $u_1 \cdots u_j$ et on pose $\rho(e_i) = \delta(0, u_1 \cdots u_{j-1})$.
L'une des difficultés sera d'énumérer les états. Une autre difficulté sera que l'on ne disposera plus d'une description aussi simple à implémenter de la fonction ρ que dans la question 16.
Le nombre d'états de l'automate sera $O\big(\max(\lambda, k|S|)\big)$, la taille du tableau des transitions $O\big(\max(\lambda, k|S|)\lambda\big)$. La complexité pourrait donc être $O\big(n + \max(\lambda, k|S|)\lambda\big)$. Si S est de taille conséquente, cela sera mieux que dans la question précédente.