TD 1: Recherche de chaînes de caractères et Expressions rationnelles

1 Recherche d'expressions rationnelles

Le but de cette partie est de concevoir un algorithme permettant de trouver toutes les occurrences d'une expression rationnelle e dans les préfixes d'un texte T en un temps $O(|e| \cdot |T|)$. On commence par construire un automate *normalisé* reconnaissant e.

Definition 1 (Automate normalisé) Un automate est dit normalisé s'il vérifie les conditions suivantes :

1. il existe un seul état initial, un seul état final, et ces deux états sont distincts ;
2. aucune flèche ne pointe vers l'état initial ni ne sort de l'état final ;
3. tout état autre que l'état final est l'origine soit d'exactement une flèche étiquetée par une lettre, soit d'au plus deux flèches étiquetées par le mot vide ϵ.

Definition 2 (Taille d'une expression rationnelle) La taille $|e|$ d'une expression rationnelle e est son nombre de symboles. Plus précisément,

$$|\epsilon| = |a| = 1 \quad \text{pour } a \in A \qquad |e + f| = |e \cdot f| = 1 + |e| + |f| \qquad |e^*| = 1 + |e|$$

Exercice 1 Montrer que pour toute expression rationnelle e, il existe un automate normalisé reconnaissant $\mathcal{L}(e)$ et dont le nombre d'états est au plus $2 \cdot |e|$.

On cherche maintenant à calculer l'ensemble des états accessibles à partir de l'état initial en lisant un texte T de manière efficace, c'est-à-dire sans construire explicitement l'automate détermiste correspondant. Soit N la taille de l'automate obtenu à l'étape précédente.

Exercice 2 Donner un algorithme `Trans`(P, a) qui, étant donnés un ensemble d'états P et une lettre a renvoie l'ensemble des états accessibles depuis P en ayant lu a. Cet algorithme devra s'exécuter en temps $O(N)$.

Exercice 3 Donner un algorithme `Epsilon`(P) qui calcule en temps également $O(N)$ l'ensemble des états accessibles par ϵ-transitions à partir de l'ensemble d'états P. On pourra tout d'abord représenter P simultanément par la liste L de ses éléments et par un tableau V de taille N.

Exercice 4 En déduire un algorithme en temps $O(|e| \cdot |T|)$ qui renvoie les préfixes de T qui appartiennent à $\mathcal{L}(e)$.

2 Automate des parties

Exercice 5 Soit $P \in \Sigma^*$ un motif de longueur m. On s'intéresse au langage L des mots ayant P pour suffixe. Montrer que ce langue est rationnel en exhibant un automate non-déterministe $\mathcal{A}$ à $m+1$ états $Q = \{0 \dots m\}$ vérifiant :

$$\mathcal{L}(0) = L$$
$$\forall 0 < k \leq m \ \mathcal{L}(k) = \{P[k+1, m]\}$$

Definition 3 Soit π^* définit par :

$$\pi^*(0) = \{0\}$$
$$\forall 0 < k \leq m \ \pi^*(k) = \{k\} \cup \pi^*(\pi(k))$$

Exercice 6 Montrer que $\pi^*(k) = \{k' \leq k \mid P[1, k'] \sqsupseteq P[1, k]\}$

Exercice 7 Soit $\mathcal{A}'$ l'automate des parties issu de $\mathcal{A}$. Montrer que l'ensemble Q' des états accessibles dans $\mathcal{A}'$ est exactement :

$$\{\pi^*(k) \mid 0 \leq k \leq m\}$$

3 Périodes d'un mot

Definition 4 On dit que p est une période de u si $\forall i \in \{1, \dots, |u| - p\}, u[i] = u[i+p]$. On note $Period(u)$ la plus petite période de u (qui existe toujours, puisque $|u|$ est toujours période de u). On dit que u est *périodique* si $Period(u) \leq |u|/2$.

Exercice 8 On considère la fonction π associée au mot u de taille n. Montrer que $Period(u) = n - \pi(n)$.

Exercice 9 Montrer que si p et q sont deux périodes d'un mot u, et si $p + q \leq |u|$, alors $pgcd(p, q)$ est aussi une période de u

3.1 Le cas des motifs auto-maximaux

On fixe un ordre total sur l'alphabet, et on note $MaxSuf(w)$ le suffixe maximal de w au sens de l'ordre lexicographique. Le mot w est dit *auto-maximal* si $MaxSuf(w) = w$.

Exercice 10 Démontrer les propriétés suivantes :

1. $|MaxSuf(w)| > |w| - Period(w)$
2. Si w est périodique, alors $Period(MaxSuf(w)) = Period(w)$;
3. Si w est auto-maximal, chacun de ses préfixes l'est.

On considère l'algorithme suivant.

```
période_naïve(w,j) :=
  Period := 1;
  pour i de 2 à j faire
    si w[i] ≠ w[i − Period] alors Period := i;
  retourner Period;
```

Exercice 11 Montrer que, lorsque w est auto-maximal, période_naïve(w,j) calcule bien $Period(w[1,j])$. Pour cela, on montrera que, pour $p = Period(w[1,i-1])$, si $w[i] \neq w[i-p]$ alors $w[i] < w[i-p]$. On montrera qu'on obtient alors une contradiction si on suppose en plus que $Period(w[1,i]) < i$.

Considérons désormais l'algorithme

```
plpam(w) :=
  Period := 1;
  pour i de 2 à |w| faire
    si w[i] < w[i − Period] alors Period := i;
    sinon si w[i] > w[i − Period] alors
      retourner (i − 1, Period);
  retourner (|w|, Period);
```

Exercice 12 Montrer que plpam(w) calcule bien le plus long préfixe auto-maximal de w et sa plus courte période.