TD corrigé langages rationnels

I Commutativité (lemme de Lévi)

Soient x et y deux mots qui commutent pour la concaténation, c'est à dire: $xy = yx$.
Montrer par récurrence sur $|x| + |y|$ que x et y sont des puissances du même mot, c'est à dire qu'il existe z tel que $x = z^k$ et $y = z^p$.
▶ Soit $H(n)$: « Si $|x| + |y| = n$ et $xy = yx$ alors x et y sont des puissances du même mot »
Si $|x| + |y| = 0$ alors $|x| = 0$ et $|y| = 0$ et $x = y = \varepsilon$. Donc $H(0)$ est vraie.
Supposons $H(k)$ vraie $\forall k \leq n$. Soient x et y tels que $xy = yx$ et $|x| + |y| = n + 1$. Quitte à inverser x et y, on peut supposer $|x| \leq |y|$. Comme $xy = yx$, x est préfixe de y donc il existe un mot u tel que $y = xu$. Alors l'équation $xy = yx$ devient $xy = xux$, d'où $y = ux$. Donc $xu = y = ux$: x et z sont deux mots qui commutent. Si $x = \varepsilon$ alors $x = y^0$, donc on peut supposer $x \neq \varepsilon$, d'où $|x| > 0$. Donc $|x| + |u| < |x| + |y|$, ce qui permet d'appliquer l'hypothèse de récurrence sur x et u: il existe z tel que $x = z^k$ et $u = z^p$. Alors $y = xu = z^{k+p}$, ce qui prouve $H(n + 1)$ et conclut la récurrence.

II Règles opératives

Dire si chacune des propositions suivantes est vraie (pour toutes expressions rationnelles e, e_1, e_2):

1. $(e^*)^* \equiv e^*$
2. $(e_1 + e_2)^* \equiv e_1^* + e_2^*$
3. $(e_1 e_2)^* \equiv e_1^* e_2^*$
4. $(e_1 + e_2)^* \equiv (e_1^* e_2^*)^*$

▶ 2. est faux car $ab \in (a + b)^*$ mais $ab \notin a^* + b^*$.
3. est faux car $abab \in (ab)^*$ mais $abab \notin a^*b^*$.
1. est vraie.
4. est vraie.

III Petites questions

1. Écrire une expression rationnelle dont le langage est l'ensemble des mots sur $\{a, b, c\}$ contenant exactement un a et un b (et un nombre quelconque de c).
   ▶ En distinguant le cas où a est avant b et le cas où b est avant a: $c^*ac^*bc^* + c^*bc^*ac^*$.

2. Montrer que le langage sur $\{0, 1\}$ des écritures en base 2 de nombres divisibles par 4 est rationnel (on omettra $<>_2$).
   ▶ C'est le langage de l'expression rationnelle $0 + 1(0 + 1)^*00$ (le nombre doit soit être 0, soit commencer par un 1 et finir par 00 en base 2).

3. Écrire une expression rationnelle dont le langage est l'ensemble des mots sur $\{a, b, c\}$ ne contenant pas de a consécutifs (aa ne doit pas apparaître).
   ▶ On peut donner $(a(b + c) + b + c)^*(a + \varepsilon)$ (un a doit être suivi d'un b ou d'un c).

4. Écrire une expression rationnelle dont le langage est l'ensemble des mots sur $\{a, b, c\}$ contenant exactement deux a et tels que tout c est précédé d'un b.
   ▶ Soit $e = (b + bc)^*$ (décrivant tous les mots sur $\{b, c\}$ dont chaque c est précédé d'un b). Alors $eaeae$ est une expression rationnelle qui convient.

5. Soit $m = m_1...m_n$ un mot sur un alphabet Σ. Montrer que l'ensemble des mots de Σ^* contenant m comme sous-mot (non forcément consécutif) est un langage rationnel.
   ▶ C'est aussi $\Sigma^* m_1 \Sigma^* m_2 ... \Sigma^* m_n \Sigma^*$ donc rationnel car concaténation de langages rationnels.

6. Si $x \in \mathbb{R}$, on note $L(x)$ l'ensemble des préfixes des chiffres de x après la virgule. Par exemple, $L(\pi) = \{\varepsilon, 1, 14, 141, 1415...\}$. En sachant que $\frac{1}{6} = 0.1666...$ et $\frac{1}{7} = 0.142857142857...$, montrer que $L(\frac{1}{6})$ et $L(\frac{1}{7})$ sont rationnels.
   ▶ $\varepsilon + 16^*$ est une expression rationnelle de langage $L(\frac{1}{6})$.

   $(142857)^*(\varepsilon + 1 + 14 + 142 + 1428 + 14285 + 142857)$ est une expression rationnelle de langage $L(\frac{1}{7})$.

7. Montrer plus généralement que $L(x)$ est rationnel si $x \in \mathbb{Q}$ (on montrera plus tard que c'est en fait une équivalence).
   ▶ Si $x \in \mathbb{Q}$, on peut écrire ses chiffres sous la forme $x = x_1, x_2 ppp...$
   Soit $Pref(m)$ l'ensemble des préfixes d'un mot m, qui est un ensemble fini si m est fini ($|Pref(m)| = |m| + 1$).
   Alors $L(x) = Pref(x_2) + x_2 p^* Pref(p)$ (un élément de $L(x)$ est soit un préfixe de x_2 soit contient x_2 suivi d'un certain nombre de p, suivi d'une partie de p.

IV Miroir d'un langage

Si $m = m_1...m_n$ est un mot, on définit son miroir $\widetilde{m} = m_n...m_1$.
Si L est un langage, on définit son miroir $\widetilde{L} = \{\widetilde{m} \mid m \in L\}$.

1. Donner une expression rationnel du miroir de $L(a(a+b)^*b)$.
   ▶ $L(b(a+b)^*a)$ (les mots commençant par b et finissant par a).

2. Soit e une expression rationnelle de langage L. Définir récursivement une expression rationnelle $\widetilde{e}$ de langage $\widetilde{L}$. Ceci montre que le miroir d'un langage rationnel est rationnel.
   ▶ (On fait ici l'abus de notation de confondre une expression rationnelle e avec son langage $L(e)$)

   (a) Si $e = a$ est une lettre: $\widetilde{e} = a$

   (b) Si $e = e_1 e_2$: $\widetilde{e} = \widetilde{e_2}\widetilde{e_1}$. En effet: $m \in \widetilde{L(e)} \iff m = \widetilde{m'}$ avec $m' \in L(e) \iff m = \widetilde{m'}$ avec $m' = m_1 m_2$ où $m_1 \in L(e_1)$, $m_2 \in L(e_2) \iff m = \widetilde{m_2}\widetilde{m_1}$ avec $m_1 \in L(e_1)$, $m_2 \in L(e_2) \iff m \in L(\widetilde{e_2}\widetilde{e_1})$.

   (c) Si $e = e_1 + e_2$: $\widetilde{e} = \widetilde{e_1} + \widetilde{e_2}$. En effet: $m \in L(\widetilde{e}) \iff m = \widetilde{m'}$ avec $m' \in L(e_1)$ ou $m' \in L(e_2) \iff m \in \widetilde{L(e_1)}$ ou $m \in \widetilde{L(e_2)}$.

   (d) Si $e = e_1^*$: $\widetilde{e} = \widetilde{e_1}^*$. Démonstration similaire aux précédentes.

3. Écrire une fonction Caml `miroir : 'a regexp -> 'a regexp` renvoyant le miroir d'une expression rationnelle. On utilisera le type suivant d'expression régulière (`L(a)` désigne une lettre `a`):

```
type 'a regexp = Epsilon | Vide | L of 'a | Somme of 'a regexp * 'a regexp
  | Concat of 'a regexp * 'a regexp | Etoile of 'a regexp;;
```

   ▶

```
let rec miroir = function
  | Vide -> Vide
  | Epsilon -> Epsilon
  | L(a) -> L(a)
  | Somme(e1, e2) -> Somme(miroir e1, miroir e2)
  | Concat(e1, e2) -> Concat(miroir e2, miroir e1)
  | Etoile(e) -> Etoile(miroir e);;
```

V Hauteur d'étoile

La hauteur d'étoile h d'une expression régulière est définie récursivement de la manière suivante:

- $h(e) = 0$ si e est $\emptyset$, ε ou une lettre.
- $h(e_1 + e_2) = \max(h(e_1), h(e_2))$.
- $h(e_1 e_2) = \max(h(e_1), h(e_2))$.
- $h(e^*) = h(e) + 1$.

1. Quelle est la hauteur d'étoile de $(ba^*b)^*$?
   ▶ $h((ba^*b)^*) = h(ba^*b) + 1 = \max(h(b), h(a^*b)) + 1$. Or $h(a^*b) = \max(h(a^*), h(b)) = \max(h(a) + 1, 0) = 1$. Donc $h((ba^*b)^*) = \max(0, 1) + 1 = 2$.
   En lisant la définition, on comprend que $h(e)$ est le nombre maximum d'étoiles imbriquées dans e.

2. Écrire la fonction h : `'a regexp -> int` en Caml.
   ▶

```
let rec h = function
  | Vide -> 0
  | Epsilon -> 0
  | L(a) -> 0
  | Somme(e1, e2) -> max (h e1) (h e2)
  | Concat(e1, e2) -> max (h e1) (h e2)
  | Etoile(e) -> 1 + h e;;
```

La hauteur d'étoile d'un langage L est la plus petite hauteur d'étoile d'une expression rationnelle e de langage L.

3. Que peut-on dire des langages de hauteur d'étoile 0?
   ▶ Ce sont exactement les langages finis.

4. Montrer que la hauteur d'étoile de $L((ba^*b)^*)$ est 1.
   ▶ $\varepsilon + b(a+bb)^*b$ a le même langage et est de hauteur d'étoile 1. De plus, comme $L((ba^*b)^*)$ est infini, toute expression rationnelle ayant ce langage doit contenir au moins une *.

VI Distance de Hamming

Soit Σ un alphabet. Si $u = u_1...u_n$ et $v = v_1...v_n$ sont deux mots de même longueur sur Σ, leur distance de Hamming est:

$$d(u,v) = |\{i \mid u_i \neq v_i\}|$$

1. Montrer que la distance de Hamming est une distance sur Σ^*.
   ▶ Soient $u = u_1...u_n, v = v_1...v_n, w = w_1...w_n$ trois mots de même taille. Si $u_i \neq w_i$ alors $u_i \neq v_i$ ou $v_i \neq w_i$ (sinon, $u_i = v_i = w_i$). D'où $d(u,v) + d(v,w) \leq d(u,w)$. $d(u,v) = d(v,u)$ et $d(u,v) = 0 \Leftrightarrow u = v$ sont facilement vérifiés.

2. Écrire une fonction `dist : 'a list -> 'a list -> int` calculant la distance de Hamming de deux mots de même longueur, sous forme de listes.
   ▶

```
let dist u v = match (u, v) with
  | [], _ -> 0
  | u1::u2, v1::v2 -> (if u1 = v1 then 0 else 1) + dist u2 v2;;
```

Étant donné un langage L sur Σ, on définit son voisinage de Hamming $\mathcal{H}(L) = \{u \in \Sigma^* \mid \exists v \in L,\ d(u,v) = 1\}$.

3. Donner une expression rationnelle du voisinage de Hamming de $L(0^*1^*)$.
   ▶ C'est l'ensemble des mots obtenus en changeant un 0 par un 1 ou inversement, c'est à dire $L(0^*10^*1^*+0^*1^*01^*)$.

4. Définir par récurrence une fonction H telle que, si e est une expression rationnelle d'un langage L sur $\Sigma = \{0,1\}$, $H(e)$ est une expression rationnelle de $\mathcal{H}(L)$.
   ▶

   (a) $H(0) = 1,\ H(1) = 0$

   (b) $H(e_1e_2) = H(e_1)e_2 + e_1H(e_2)$: modifier une lettre de $u = u_1u_2 \in L(e_1e_2)$ revient à modifier une lettre de u_1 ou un lettre de u_2.

   (c) $H(e_1 + e_2) = H(e_1) + H(e_2)$.

   (d) Si $e = e_1^*$: $H(e_1^*) = e_1^*H(e_1)e_1^*$.

5. Écrire la fonction H précédente en Caml.
   ▶

```
let rec ham e = match e with
  | Vide -> Vide
  | Epsilon -> Epsilon
  | L(a) -> L(1-a)
  | Somme(e1, e2) -> Somme(ham e1, ham e2)
  | Concat(e1, e2) -> Somme(Concat(ham e1, e2), Concat(e1, ham e2))
  | Etoile(e') -> Concat(e, Concat(ham e', e));;
```

VII Code de Prüfer

Soit $\mathcal{T}_n$ l'ensemble des arbres (vus comme des graphes connexes sans cycle) sur l'ensemble de sommets $V = \{0, 1, ..., n-1\}$. On aimerait avoir un codage de ces arbres, c'est à dire une bijection de $\mathcal{T}_n$ vers un certain langage simple sur l'alphabet V.

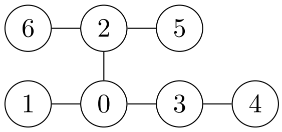
1. Soit $f : \mathcal{T}_n \longrightarrow V^{n-1}$ telle que $f(T) = p_1...p_{n-1}$ où p_i le père du sommet i (en enracinant T en 0).
   f est-elle injective? Surjective?

Si T est un arbre, son codage de Prüfer est obtenu à partir d'un mot vide m de la façon suivante:

Codage de Prüfer

Tant que T a au moins 2 sommets:
  Soit v la feuille de numéro minimum de T.
  Ajouter le voisin de v à m.
  Supprimer v de T.

2. Quel est le code de Prüfer de l'arbre ci-dessous?



3. Montrer que $g : \mathcal{T}_n \longrightarrow V^{n-2}$ qui à un arbre associe son code de Prüfer est une bijection.
4. Donner une formule simple pour $|\mathcal{T}_n|$ (c'est la formule de Cayley).
5. Comment pourrait-on générer un arbre uniformément au hasard dans $\mathcal{T}_n$?
6. (Pour les plus motivés) Écrire des fonctions Caml pour calculer le code de Prüfer d'un arbre et pour le décoder.