Corrigé du devoir à la maison du 24 mars

Deux questions délicates de TD sont également traitées en page 4.

Un pseudo fascicule de définitions et de résultats vous a par ailleurs été distribué, il couvre surtout le programme de l'examen de juin.

1. Monoïdes

1.1. Un monoïde est un ensemble muni d'une loi interne (partout définie, associative) et d'un élément neutre.

1.2. Un monoïde libre est monoïde avec une "base". C'est-à-dire que ce monoïde est *engendré* par un ensemble d'éléments *indépendants* (=dépourvus de relations entre eux). ("Base" et "libre" sont donc à rapprocher des notions de base ou de liberté dans un espace vectoriel.) On a ainsi une "factorisation" (=une décomposition) de chaque élément du monoïde libre en élèments de cette "base". On peut de plus alors définir la longueur d'un élément. Parfois, cette base est qualifiée d'*alphabet*, les générateurs (= les éléments de la base) sont appelés *lettres* et les éléments du monoïde libre sont appelés *mots*, l'opération associative est appelée *concaténation* et l'élément neutre est appel*mot vide*.

1.3. Exemple de monoïde libre : les entiers positifs ou nul avec l'addition comme opération (elle est bien associative et 0 est l'élément neutre, la base est $\{1\}$). Exemple de monoïde non libre : les entiers relatifs avec l'addition comme opération (la base est $\{-1,+1\}$ mais cette base n'est pas "libre" car $(+1)+(-1)=0$).

Deux autres exemples seraient
monoïde libre : l'ensemble E_1 des mots contenants des a ou des b, muni de la concaténation comme loi interne et du mot vide comme élément neutre. La "base" de cet ensemble est $\{a,b\}$, E est alors parfois noté $\{a,b\}^*$;
monoïde non libre : l'ensemble E_2 des mots contenants des a ou des b, muni de la concaténation comme loi interne et du mot vide comme élément neutre et pour lequel on identifie les mots du type $uabbv$ avec le mot uv. Cette fois-ci $\{a,b\}$ n'est plus une base car on a la relation $abb=\epsilon$.

1.4. Il y avait 143 mots à énumérer (travail fastidieux prenant 30 minutes avec les très nécessaires vérifications). Une présentation possible de tous ces mots peut se faire sous la forme d'un arbre de possibilités, pour une présentation plus "linéaire", on peut calculer les produits (jusqu'a 10) et faire l'union, mais cela va peut-être un peu plus vite de s'apercevoir que L^* représente les mots où un a est aussitôt suivi d'un b et donc les mots de longueur <10 sont {ϵ, b, ab, bb, bab, bbb, abb, abab, abbb, babb, bbab, bbbb, ababb, abbab, abbbb, babab, babbb, bbabb, bbbab, bbbbb, ababab, ababbb, abbabb, abbbab, abbbbb, bababb, babbab, babbbb, bbabab, bbabbb, bbbabb, bbbbab, bbbbbb, abababb, ababbab, ababbbb, abbabab, abbabbb, abbbabb, abbbbab, abbbbbb, bababab, bababbb, babbabb, babbbab, babbbbb, bbababb, bbabbab, bbabbbb, bbbabab, bbbabbb, bbbbabb, bbbbbab, bbbbbbb, ababab ab, abababbb, ababbabb, ababbbab, ababbbbb, abbababb, abbabbab, abbabbbb, abbbabab, abbbabbb, abbbbabb, abbbbbab, abbbbbbb, babababb, bababbab, bababbbb, babbabab, babbabbb, babbbabb, babbbbab, babbbbbb, bbababab, bbababbb, bbabbabb, bbabbbab, bbabbbbb, bbbababb, bbbabbab, bbbabbbb, bbbbabab, bbbbabbb, bbbbbabb, bbbbbbab, bbbbbbbb, abababab b, ababab bab, abababbbb, ababbabab, ababbabbb, ababbbabb, ababbbbab, ababbbbbb, abbababab, abbababbb, abbabbabb, abbabbbab, abbabbbbb, abbbababb, abbbabbab, abbbabbbb, abbbbabab, abbbbabbb, abbbbbabb, abbbbbbab, abbbbbbbb, babababab, bababab bb, bababbabb, bababbbab, bababbbbb, babbababb, babbabbab, babbabbbb, babbbabab, babbbabbb, babbbbabb, babbbbbab, babbbbbbb, bbabababb, bbababbab, bbababbbb, bbabbabab, bbabbabbb, bbabbbabb, bbabbbbab, bbabbbbbb, bbbababab, bbbababbb, bbbabbabb, bbbabbbab, bbbabbbbb, bbbbababb, bbbbabbab, bbbbabbbb, bbbbbabab, bbbbbabbb, bbbbbbabb, bbbbbbbab, bbbbbbbbb}.

Certains auront peut-être remarqué que le nombre a_n de mots de L^* de longueur n donne une suite (a_n) qui vérifie la récurrence $a_{n+2}=a_{n+1}+a_n$ (les premiers termes sont $(1,1,2,3,5,8,13,21,34,55,...)$), il s'agit là de la suite de Fibonacci. La section 5 (Analyse et théorie des langages) reviendra sur ce type de considérations.

2. Grammaires

2.1. L_1={les mots ne contenant pas deux a consécutifs} est engendré par

$$S\to\epsilon|a|B|BaBb|BaBba|,\quad B\to bB|\epsilon$$

On engendre ainsi le mot vide, le mot a, les mots qui ne contiennent que des b, puis les mots qui contiennent des a (nécessairement suivi d'un b) et qui se terminent, soit par b, soit par a.

L_2={les mots contenant autant de a que de b} est engendré par

$$S\to aSbS|bSaS|\epsilon$$

On a ainsi "apparié" les occurences de a et b pour assurer l'équilibre.

L_3 ={les mots ne contenant que des a (au moins un) puis que des b (au moins un)} est engendré par

$$S \to aAbB, \quad A \to aA|\epsilon, \quad B \to bB|\epsilon$$

L_4={les mots ne contenant que des a puis que des b, avec "nombre de b" > "nombre de a" > 0} est engendré par

$$S \to aSb|bS|\epsilon$$

L_5={les mots dont la parité du nombre de a égale celle du nombre de b} est engendré par

$$S \to aT|bT|\epsilon, \quad T \to aS|bS$$

L'idée de base consiste á partir du mot vide (dont la parité de nombre a égale celle du nombre de b) et á observer que si j'ajoute un a ou un b, la parité n'est plus égale, et alors si j'ajoute un a ou un b, la parité redevient égale. On voit ainsi que tous les mots de longueur paire appartiennent á L_5 (et seulement eux!).

2.2. L_1 peut être engendré par la grammaire régulière suivante :

$$S \to bB|bA|aA|\epsilon, \quad B \to bB|\epsilon, \quad A \to bB|bD, \quad D \to aA|\epsilon$$

L_3 peut être engendré par la grammaire régulière suivante :

$$S \to aA, \quad A \to aA|bB, \quad B \to bB|\epsilon$$

La grammaire de L_5 est déjà réguliìere.
(Expressions rationnelles associées : $L_1 = b^*(ab^+)^*(\epsilon + a)$, $L_3 = a^+b^+$, $L_5 = (aa + ab + ba + bb)^*$)

2.3. À l'aide du lemme de l'étoile (voir preuve, page 4), on va prouver que $L_4 = \{a^nb^{n+k}, n \geq 1, k > 0\}$ n'est pas régulier. Supposons que L_4 est régulier, regardons alors la décomposition $u = u_1vu_2$ (pour u un mot de L_4 "assez long") qui nous est donnée par application du lemme de l'étoile. On devrait avoir $u_1v^iu_2 \in L_4$ pour tout $i \in \mathbf{N}$. Or ceci est clairement impossible si v ne contient que des a, ou si v contient des a et des b.

Donc les décompositions fournies ne peuvent donner que des v qui ne contiennent que des b.

Or il existe dans L_4 des mots avec un préfixe contenant autant de a consécutifs que l'on veut.

Ceci implique donc qu'avant même que l'on commence à lire des b dans l'automate, on est déjà passé deux fois par le même état. Ainsi, on a bien l'existence d'un facteur v qui ne contient que des a, ce qui contredit le point précedent et donc ceci implique que l'on ne pouvait appliquer le lemme de l'étoile, et que le langage L_4 n'est donc pas régulier.

3. Expressions régulières

3.1. On commence par simplifier un peu l'expression : (a^*b^*) contient a et b, donc $(a^*b^*)^* = \{a,b\}^*$. Ainsi

$$(a^*b^*)^*b + (ba)^* = \{a,b\}^*b + (ba)^*$$

et il s'agit donc de reconnaître tous les mots finissants par b et toutes les suites de ba. La construction de l'automate ne pose aucun problème.

3.2. Enlevons la transition entre l'état 4 et l'état 1 et regardons l'automate A' ainsi défini.

Par disjonction des cas, le langage L' reconnu par cet automate est

$$L' = a(ba)^*b + a(ba)^*ba + b(ab)^*a + b(ab)^*ab.$$

Il est clair que

$$\mathcal{L}(A) = (L'a)^*L' = ((a(ba)^*(b + ba) + b(ab)^*(a + ab))a)^*(a(ba)^*(b + ba) + b(ab)^*(a + ab)).$$

Diverses autres formules équivalentes (à une factorisation près, etc.) été aussi acceptables.

3.3. Impossible de donner une telle expression car le langage n'est pas régulier.

En effet, c'est une grammaire équivalente à

$$A \to aAbb|aab|ab^*A|\epsilon$$

qui est identique, à des changements de variables prets, à

$$A \to cAd|eA|f$$

et si on fait maintenant les substitutions ($e \leftarrow \epsilon$ et $f \leftarrow \epsilon$), on ne modifie pas la régularité du langage; or le langage engendré par

$$A \to cAd|\epsilon A|\epsilon$$

est $\{c^nd^n), n \in \mathbf{N}\}$, bien connu pour ne pas être régulier (démontré en TD). CQFD !

La transformation suivante a pour trace $aabbb$:

$$A \to aBb \to Ab \to aC \to bC \to A \to \epsilon$$

La grammaire est ambigüe car on peut trouver un autre arbre de dérivation, celui qui correspond à la transformation

$$A \to aC \to Aa \to C \to bC \to bC \to bC \to A \to \epsilon$$

4. Automates

4.1. Il suffit d'enlever l'état 0.

4.2. [dessin à faire] Application de l'algorithme vu en TD.

4.3. On ne peut pas faire mieux que l'automate donné en 4.1.

5. Analyse et théorie des langages

5.1.

$$S \to 0S|1S|\epsilon \Rightarrow S(z) = \frac{1}{1-2z}$$

Dans le même genre, la série génératrice de $(ab, b)^* = \cup_{n\in\mathbf{N}}\{a, ab\}^n$ (cf. question 1.4) est $\sum(z+z^2)^n = \frac{1}{1-(z+z^2)}$.

5.2. La grammaire $S \to aSbS|\epsilon$ est non ambigüe et génère les mots de Dyck. La série génératrice est donnée par

$$S \to aSbS|\epsilon \Rightarrow S(z) = z^2S(z)^2 + 1 \Rightarrow L(z) = S(z) = \frac{1-\sqrt{1-4z^2}}{2z^2}$$

La division par z^2 n'est pas problématique en $z = 0$, car on a en fait une forme indéterminée $0/0$ qui s'avère être $= 1$ (en utilisant des développements limités par exemple). En revanche, la série a une singularité (du type $\sqrt{0}$) quand $1 - 4z^2 = 0$ donc quand $z = 1/2$ qui est donc le rayon de convergence, on en déduit (formule de Hadamard : rayon de convergence $= 1/\text{lim sup } a_n^{1/n}$) que les coefficients de la série croissent comme 2^n. C'est-à-dire qu'il y a environ 2^n mots de Dyck de longueur n (pour n pair, car pour n impair, il y en a bien sûr 0).

5.3. D'après le théorème de Chomsky-Schützenberger, L_2 n'est pas reconnaissable car il engendré par une grammaire non ambigüe et que sa série génératrice est algébrique non rationnelle :

$$S \to aSbS|bSaS|\epsilon \Rightarrow S(z) = 2z^2S(z)^2 + 1 \Rightarrow L(z) = S(z) = \frac{1-\sqrt{1-8z^2}}{4z^2}$$

6. Automates et chaînes de Markov

6.1. [dessin à faire]

6.2. [dessin à faire]

Comme les probabilités d'arrivées dans les différents états au bout de trois coups sont identiques, tout se joue après.

Soit M_1 la matrice donnant les transitions entre les 8 états (après 3 coups) pour PPP final et M_2 la matrice pour PFP final. [dessin à faire]

Il faut alors utiliser un logiciel de calcul formel pour calculer $\sum M_1^n = (Id - M_1)^{-1}$ et $\sum M_2^n = (Id - M_2)^{-1}$.

La somme des éléments de la i-ième ligne de l'inverse de $Id - M$ donne donc le nombre moyen de coups a faire avant d'atteindre l'état correspondant.

On reviendra en TD sur cet exercice (dont la dernière question est totalement hors programme).

7. Compilation

Compiler, c'est transformer une “phrase” (un programme) d'un certain langage en “quelque chose” de compréhensible par le microprocesseur.

- L'analyse lexicale (scanning) consiste à lire un par un les lettres (= caractères, symboles...) afin de reconnaître des mots du langage, et donc de découper la phrase initiale en suite de mots.
- L'analyse syntaxique (parsing) répond à la question : cette suite de mots ainsi obtenue est-elle “grammaticalement” correcte ? C'est-à-dire, peut-on trouver une dérivation qui conduit à cette suite de mots ?
- L'analyse sémantique s'attache à détecter des incohérences de sens.
- L'optimisation consiste à simplifier l'arbre de dérivation ainsi obtenu.
- La génération de code consiste à associer des mots d'un nouveau langage (traduction en langage machine) aux mots précédemment reconnus.
- L'exécution est alors l' “interprétation” par le micro-processeur du code ainsi généré.

8. TD1 Exercice 3. Question c)

On rappelle qu'avec les notions de cette question, $bin(u)$ est le nombre d'opération du mot u et que $D(u)$ est le nombre de "découpage" du mot u.

Nous allons faire une récurrence **forte** sur $bin(u)$.

• [initialisation de la récurrence]
Quand $bin(u) = 0$, on a donc utilisé ni la règle 1 ni la règle 2, on n'a donc aucune ambigüité (l'arbre de dérivation de u est en fait "linéaire") et on a bien
$$F(u) = Amb_G(u) = 1.$$

• [héridité **forte**] Supposons que $F(u) = Amb_G(u)$ soit vérifiée pour les u tels que $bin(u) \leq n$ (**inférieurs** ou égal, j'insiste), nous voulons montrer que $F(v) = Amb_G(v)$ pour tous les v tels que $bin(v) = n + 1$.

Or un tel v peut bien sûr s'écrire sous la forme $v = u_1$ *opérateur* u_2, et donc u_1 et u_2 sont tels que $bin(u_1) \leq n$ et $bin(u_2) \leq n$.

Il y a tous les $D(u)$ comme découpage possible, ceci réflète tout à fait les différents arbres de dérivations possibles : quand on regarde la racine, on a soit $S_1 \to S_2 \times S_3$ soit $S_1 \to S_2 + S_3$ avec S_2 qui va se dériver en u_1 et S_3 qui va se dériver en u_2.

Ainsi
$$Amb_G(u) = \sum_{(u_1,u_2)\in D(u)} Amb_G(u_1)Amb_G(u_2) = \sum_{(u_1,u_2)\in D(u)} F(u_1)F(u_2) = F(u).$$

(La deuxième égalité étant due à l'hypothèse de récurrence forte.)

On a bien démontré $F = Amb_G$.

9. TD2 Exercice 2

Lemme 1 (Lemme de l'étoile ou "pumping lemma"). *Si L est un langage régulier (c'est-à-dire $L = \mathcal{L}(G)$ pour une grammaire G régulière) alors :*
il existe un entier naturel $N > 0$ tel que, pour tout $u \in L$ vérifiant $|u| \geq N$, on ait trois mots u_1, v, u_2 satisfaisant les conditions :
1) $u = u_1vu_2$
2) $|v| \neq 0$
3) pour tout entier naturel i : $u_1v^iu_2 \in L$.

Proof. Pour un langage fini L (i.e. qui contient un nombre fini de mots), le lemme de l'étoile est "bêtement" vérifié (car, en choissisant N:=1+"longueur du plus long mot de L", on a aucun mot de longueur $\geq N$).

Pour un langage régulier et infini L (i.e. qui contient un nombre infini de mots), considérons un automate A tel que $L = \mathcal{L}(A)$ (il existe un tel automate car **Reg=Rec**, tout langage **rég**ulier est **rec**onnu par un certain automate). Notons m le nombre d'états de A. Considérons maintenant un mot $u \in L$ de longueur $> m$ (il existe forçément un tel mot car le langage est infini), il existe donc un chemin dans A qui part d'un état initial, qui passe obligatoirement (au moins) deux fois par le même état x (car A n'a que m états), et qui termine dans un état final.
Notons u_1 la trace du chemin entre l'état initial et la première fois que l'on est passé par x.
Notons u_2 la trace du chemin entre la dernière fois que l'on est passé par x et l'état final.
Notons v la trace du chemin entre la première et la dernière fois que l'on est passé par x.
On a bien $u = u_1vu_2$ et $v \neq \epsilon$. De plus, $\forall i, u_1v^iu_2$ est bien un mot de L car il correspond à un chemin partant d'un état initial et finissant dans un état final.

Il suffit donc de prendre $N = m + 1$ et on a prouvé le lemme de l'étoile.

□