Syllabus de Théorie des langages.

Gérard H. E. Duchamp*

8 avril 2012

08-04-2012 07:47

Table des matières

1 Calcul du langage reconnu par un AF

1.1 Automates finis et structures de transition

Rappelons qu'un **AF** (automate fini est une machine définie par un 5-uplet $\mathcal{A} = (Q, A, \bullet, I, F)$ où

– Q est un ensemble fini d'états

* Prière de me signaler les erreurs.

– A est un alphabet fini
– $\bullet : Q \times A \mapsto \mathcal{P}(Q)$ est une application
– $I \subseteq Q$ est l'ensemble des entrées (i.e. états initiaux)
– $F \subseteq Q$ est l'ensemble des sorties (i.e. états finaux)

Note 1.1 *Si on oublie les états initiaux et finaux, c'est une* structure de transition. *On appelle donc* structure de transition *la donnée d'un triplet* $(Q,A,\bullet)$ *avec les caractéristiques du paragraphe précédent.*

La structure de transition est équivalente à le donnée d'un graphe orienté étiqueté $T \subset Q \times A \times Q$. Les arêtes sont donc des triplets $h = (p,a,q)$ l'état p (resp. q, la lettre a) s'appelle l'origine (resp. l'extrémité, l'étiquette) de h. On présentera donc un automate par un quintuplet (Q,A,T,I,F). Cette manière de faire facilite la traduction des données en la *représentation linéaire* de l'automate qui est celle que l'on implémente (voir T.D.). Un *chemin* c dans le graphe de transition est une suite d'arêtes $h_i = (p_i,a_i,q_i);\ i = 1..n$ telles que, pour tout $i < n$, on ait $q_i = p_{i+1}$ (les arêtes s'enchaînent). L'étiquette de c est le mot $a_1a_2\cdots a_n$. Le langage reconnu par un automate $\mathcal{A}$, noté $\text{Ł}(\mathcal{A})$, est l'ensemble des étiquettes des chemins qui mènent d'un état initial à un état final. On verra au paragraphe suivant que ce langage peut se calculer *matriciellement* à l'aide de la représentation linéaire de l'automate.

Définition 1.2 *La représentation linéaire d'un automate* $\mathcal{A} = (Q,A,T,I,F)$ *est la donnée des matrices suivantes associées à* $\mathcal{A}$.

– *Le vecteur initial* $V_I \in \{0,1\}^{1\times Q}$ *(c'est un vecteur ligne, en général* $Q = [1..n]$*) défini par* $V_I(q) = [q \in I]$ *(symbole d'Iverson, on rappelle que* $[P] = 1$ *si* P *est vraie et* 0 *sinon). que*
– *Le vecteur final* $V_F \in \{0,1\}^{Q\times 1}$ *(c'est un vecteur colonne, en général* $Q = [1..n]$*) défini par* $V_F(q) = [q \in F]$.
– *Pour chaque* $a \in A$, *une matrice* $M(a) \in \{0,1\}^{Q\times Q}$ *définie par* $M(a)[q_1,q_2] = [(q_1,a,q_2) \in T]$

En général on note, par abus de langage, I et F au lieu de V_I et V_F.

1.2 Comportement d'un automate : une fonction sur les mots

Dans ce chapitre, nous nous intéresserons plus spécialement aux fonctions sur A^* (A est un alphabet). C'est à dire, en ce qui concerne les sysèmes (informatiques ou électroniques) aux machines qui acceptent un mot en entrée et retournent un coefficient (un scalaire: un nombre, un booléen, un réel, un complexe ou même une matrice) en sortie.

$$w \to \boxed{\text{MACHINE}} \to M(w) \tag{1}$$

Ici on utilisera les scalaires de $B = \{0,1\}$ (les booléens) et $\mathbb{N}$ (les entiers naturels).
Dans le premier cas, on exprime seulement si le mot w est reconnu ou non. Dans le

deuxième, on compte les chemins réussis d'étiquette w.
On a le lemme suivant.

Lemme 1.3 *Posons, pour $w = a_1 a_2 \cdots a_n$, $M(w) = M(a_1)M(a_2)\cdots M(a_n)$ (produit matriciel). Alors, on a*

$$[w \in Ł(\mathcal{A})] = IM(w)T. \tag{2}$$

Preuve — Elle se fait en remarquant, par récurrence que
$M(w)[q_1,q_2]$ = [il existe un chemin d'étiquette w qui joint q_1 à q_2]. □

Ce lemme permet d'exprimer $Ł(\mathcal{A})$ de façon synthétique
Proposition 1.4 *Le langage reconnu par un automate $\mathcal{A}$ est bien*

$$Ł(\mathcal{A}) = \sum_{w \in A^*} (IM(w)T)w \tag{3}$$

1.3 Automate des parties

Soit $\mathcal{A} = (Q,A,\bullet\ ,I,F)$, un automate quelconque. On définit l'automate des parties $2^{\mathcal{A}}$ par

1. États : 2^Q (ou $\mathcal{P}(Q)$, l'ensemble des parties de Q)
2. Alphabet : le même (soit A)
3. $\hat{\bullet}$, donné par $P\hat{\bullet}\ x = \cup_{q\in P}\ q \bullet\ x$
4. $I = I$ (mais cette fois-ci c'est un état)
5. $F = \{P \in 2^Q | P \cap F \neq \emptyset\}$

On peut montrer que ce nouvel automate est un AFDC et que son comportement est exactement le même que celui de l'automate de départ.

1.4 Équivalence avec un AF d'un automate à ϵ-transitions

1.4.1 Généralités

Dans un automate à ϵ-transitions, on rajoute un symbole supplémentaire ϵ qui sera éliminé dans les étiquettes des chemins (ainsi l'étiquette d'un chemin est toujours la concaténation de ses étiquettes).
Le langage reconnu par un tel automate est toujours

$$\sum_{w \text{ est l'étiquette d'un chemin réussi de } \mathcal{A}} w \tag{4}$$

1.4.2 Suppression des ϵ-transitions

Les chemins réussis d'étiquette $w = a_1 a_2 \cdots a_n$ ont pour étiquette (avant supression des ϵ) $\epsilon^{k_0} a_1 \epsilon^{k_1} a_2 \epsilon^{k_2} \cdots a_n \epsilon^{k_n}$. Ceci montre que si $M(\epsilon^*)$ désigne la matrice de la clôture reflexive-transitive des ϵ-transitions, le nouvel autmate dont les éléments matriciels sont

- $I' = IM(\epsilon^*)$

– $M'(a) = M(a)M(\epsilon^*)$
– $F' = F$.

Proposition 1.5 *L'automate (sans ϵ-transitions) $\mathcal{A}'$ reconnait le même langage que $\mathcal{A}$.*

1.4.3 Application des ϵ-transitions

Les automates à ϵ-transitions sont commodes pouur construire des automates qui reconnaissent létoile et le produit de deux langages reconnus par un (ou deux) automates donnés.

2 Etoile d'une matrice

On va ici considérer des matrices (carrées) de langages. Le produit se fait comme d'habitude : si $U = (L_{ij})_{1\leq i,j\leq n}$ et $V = (M_{ij})_{1\leq i,j\leq n}$ on a $UV = (N_{ij})_{1\leq i,j\leq n}$ avec

$$N_{ij} := \sum_{1\leq k\leq n} M_{ik}N_{kj} \tag{5}$$

Rappel. — Dans $U = (L_{ij})_{1\leq i,j\leq n}$, on a i qui est l'adresse de ligne, j l'adresse de colonne. Ainsi une matrice 3×3 se dispose

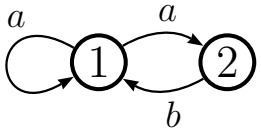
$$\begin{pmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \tag{6}$$

Définition 2.1 *La* matrice de transtition "lettres" *(ou* matrice-lettres *tout court) d'une structure de transition est une matrice de langages (qui sont des sous-alphabets). C'est la matrice de format $Q \times Q$ (ses lignes et ses colonnes sont indexées par Q)*

$$T := \Big(\sum_{q_2\in q_1.a} a \Big)_{q_1,q_2\in Q} \tag{7}$$

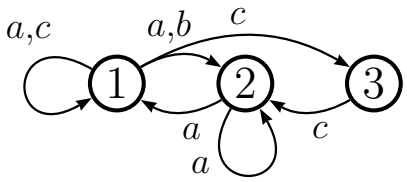
étant entendu (par convention générale) que la somme est nulle s'il n'y pas d'arête entre q_1 et q_2.

Exemple 2.2 *1) La petite structure de transition suivante:*



admet pour matrice-lettres $\begin{pmatrix} a & b \\ a & 0 \end{pmatrix}$.

admet pour matrice-lettres $\begin{pmatrix} a+c & a+b & c \\ a & a & 0 \\ 0 & c & 0 \end{pmatrix}$.



2.1 Étoile d'une matrice-lettres

Les puissances de la matrice-lettres d'une structure de transition ont une propriété remarquable.

Proposition 2.3 *Soit T, la matrice-lettres d'une structure de transition $\mathcal{T} = (Q,A,\bullet)$. Pour toute paire d'états (q_1,q_2), on a*

$$(T^k)[q_1,q_2] = \{w \in A^k | q_1.w = q_2\} \tag{8}$$

autrement dit, l'entrée d'adresse $[q_1,q_2]$ de la puissance T^k est l'ensemble des mots de longueur k qui font passer de q_1 à q_2.

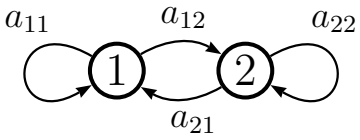


FIG. 1 – *Structure de transition générique sur 2 états*

3 Fonctions sur les mots

Dans ce chapitre, nous nous intéresserons plus spécialement aux fonctions sur A^* (A est un alphabet). C'est à dire, en ce qui concerne les sysèmes (informatiques ou électroniques) aux machines qui acceptent un mot en entrée et retournent un coefficient (un scalaire: un nombre, un booléen, un réel, un complexe ou même une matrice) en sortie.

$$w \rightarrow \boxed{\text{MACHINE}} \rightarrow M(w) \tag{9}$$

On appellera *comportement* de la machine cette fonction $A^* \rightarrow K$ (K est l'ensemble des scalaires, voir chapitre (**??**)) et deux machines seront dites équivalentes ssi elles définisent la même fonction.

On peut composer les machines à l'aide des fonctions classiques (additionneurs, multiplieurs).

$$\begin{array}{cccccc} & \nearrow & \boxed{\text{MACHINE 1}} & \searrow & & \\ w & \rightarrow & \boxed{\text{MACHINE 2}} & \rightarrow & \bigoplus & \rightarrow M1(w) + M2(w) \end{array}$$

$$\begin{array}{cccccc} & \nearrow & \boxed{\text{MACHINE 1}} & \searrow & & \\ w & \rightarrow & \boxed{\text{MACHINE 2}} & \rightarrow & \bigotimes & \rightarrow M1(w) \times M2(w) \end{array}$$

c'est le produit (ou la somme) ponctuel(le) des fonctions correspondantes.
Il y a aussi un autre type de produit qui est très utile (nous verrons qu'il généralise la concaténation et qu'il éclaire les opérations sur les parties, c'est le produit défini par le formule

$$M1 * M2(w) = \sum_{uv=w} M1(u)M2(v) \tag{10}$$

il peut être réalisé par le système suivant

$$\begin{array}{rccccccc} & \nearrow & u \to & \boxed{\text{MACHINE 1}} & \searrow & & \\ w = uv & \to & v \to & \boxed{\text{MACHINE 2}} & \to & \bigotimes & \bigoplus_{uv=w} M1(u)M2(v) \end{array}$$

3.1 Quelques exemples de machines

Toutes les machines considérées ici acceptent des mots en entrée et retournent des coefficients en sortie. Tout d'abord quelques machines de type "compteur".

Exemple 1: Longueur d'un mot. — C'est une machine (ou un programme) qui lit un mot de gauche à droite (ou de droite à gauche peu importe ici) et qui incémente un compteur de +1 à chaque lettre (le compteur est initialisé à 0).
Le résultat est la longueur du mot.
Lorsque le mot est vide le compteur rest bien à son initialisation et le résultat est 0 (longueur du mot vide).

Exemple 2: Nombres d'occurences d'une lettre. — Soit $A = \{a,b\}$. C'est une machine (ou un programme) qui lit un mot de gauche à droite (ou de droite à gauche peu importe ici) et qui incémente un compteur de +1 à chaque lecture de a (le compteur est initialisé à 0).
Le résultat est le nombre d'occurences de a.
Lorsque le mot est vide le compteur est bien à son initialisation et le résultat est 0 (longueur du mot vide).

Cette machine peut se réaliser matricellement. On pose

$$M(a) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix};\ M(b) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix};\ I = \begin{pmatrix} 1 & 0 \end{pmatrix};\ T = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{11}$$

puis $M(w) = M(a_1a_2\cdots a_n) = M(a_1)M(a_2)\cdots M(a_n)$ le résultat de la lecture d'un mot w est $IM(w)T$.
Montrons que cette machine compte bien le degré partiel en a

Exercice 3.1 *Montrer que cette machine compte bien le degré partiel en a.*

Dans la suite, une fonction $f : A^* \to K$ pourra aussi être noée $\sum_{u\in A^*} f(u)u$ (notation sommatoire). Cette notation permet de manipuler les fonctions comme des séries et les parties comme des sommes de mots.

4 Systèmes et Calcul

4.1 Introduction

Exemples d'automates booléens, stochastiques, de comptage, de plus courts chemins. Les semi-anneaux associés sont : $\mathbb{B}$, $\mathbb{R}_+$,$\mathbb{N}$,$([0, +\infty],min,+)$.

4.2 Description de la structure d'automate

4.2.1 Graphe pondéré

L'élément de base de ces graphes est la flêche $A = q_1 \stackrel{a|\alpha}{\to} q_2$ avec $q_i \in Q$, $a \in \S$, $\alpha \in k$ où Q est un ensemble d'états, $\S$ un alphabet et k, un semi-anneau[1]. Pour un tel objet, on définit, selon les conventions générales de la théorie des graphes,

- $t(A) := q_1$ ("tail": queue, source, origine)
- $h(A) := q_2$ ("head" tête, but, extrémité)
- $l(A) := a$ ("label" étiquette)
- $w(A) := \alpha$ ("weight" poids).

Un *chemin* est une suite d'arêtes $c = A_1 A_2 \cdots A_n$ (c'est un mot en les arêtes et sa longueur est n) telle que $h(A_k) = t(A_{k+1})$ pour $1 \leq k \leq n-1$ pour un tel chemin $t(c) = t(A_1)$, $h(c) = h(A_n)$, $l(c) = l(A_1)l(A_2)\cdots l(A_n)$ (concaténation), $w(c) = w(A_1)w(A_2)\cdots w(A_n)$ (produit dans le semi-anneau).
Par exemple pour le chemin de longueur 3 suivant ($k = \mathbb{N}$),

$$u = p \stackrel{a|2}{\to} q \stackrel{b|3}{\to} r \stackrel{c|5}{\to} s \tag{12}$$

on a $t(u) = p$, $h(u) = s$, $l(u) = abc$, $h(u) = 30$.
Le poids d'un ensemble de chemins de même source, but et étiquette est la somme des poids des chemins de cet ensemble. Ainsi, si

$$\mathbf{q}1 \quad \begin{array}{c} \stackrel{u|\alpha}{\to} \\ \stackrel{u|\beta}{\to} \end{array} \quad \mathbf{q}2 \tag{13}$$

le poids de cet ensemble de chemins est $\alpha + \beta$. On a donc que les poids se multiplient en série et s'aditionnent en parallèle. Les diagrammes suivants montrent la nécessité des axiomes de semi-anneau.

Diagramme	Identité	Nom
$\begin{array}{c} \stackrel{a|\alpha}{\to} \\ p \stackrel{a|\beta}{\to} q \\ \stackrel{a|\gamma}{\to} \end{array}$	$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$	Associativité de $+$
$p \begin{array}{c} \stackrel{a|\alpha}{\to} \\ \stackrel{a|\beta}{\to} \end{array} q$	$\alpha + \beta = \beta + \alpha$	Commutativité de $+$
$p \begin{array}{c} \stackrel{a|\alpha}{\to} \\ \stackrel{a|0}{\to} \end{array} q$	$\alpha + 0 = \alpha$	Élément neutre (droite) de $+$
$p \stackrel{a|0}{} q \stackrel{b|\beta}{\to} r$	$0 + \beta = \alpha$	Élément neutre (gauche) de $+$
$p \stackrel{a|\alpha}{\to} q \stackrel{b|\beta}{\to} r \stackrel{c|\gamma}{\to} s$	$\alpha(\beta\gamma) = (\alpha\beta)\gamma$	Associativité de $\times$
$p \begin{array}{c} \stackrel{a|\alpha}{\to} \\ \stackrel{a|\beta}{\to} \end{array} q \stackrel{b|\gamma}{\to} r$	$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$	Distributivité (droite) de $\times$ sur $+$
$p \stackrel{a|\alpha}{\to} q \begin{array}{c} \stackrel{b|\beta}{\to} \\ \stackrel{b|\gamma}{\to} \end{array} r$	$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$	Distributivité (gauche) de $\times$ sur $+$
$p \stackrel{a|\alpha}{\to} q \stackrel{b|1_k}{\to} r$	$\alpha \times 1_k = \alpha$	Élément neutre (droite) de $\times$
$p \stackrel{a|1_k}{\to} q \stackrel{b|\beta}{\to} r$	$1_k \times \beta = \beta$	Élément neutre (gauche) de $\times$

1. Nous verrons plus bas que les axiomes de la structure de semi-anneau sont contraints par la définition même du système de transitions ainsi obtenu.

4.2.2 Structure et comportement des automates

Un automate à poids ou pondéré ("automaton with weights") est la donnée de trois éléments vectoriels (I,M,T):

$$\begin{cases} \bullet \text{ Un vecteur d'entrée } I \in k^{1\times Q} \\ \bullet \text{ Une famille (indexée à A) de matrices de transition } M : A \to k^{Q\times Q} \\ \bullet \text{ Un vecteur de sortie } T \in k^{Q\times 1} \end{cases}$$

La donnée des transitions (M) est équivalente à celle d'un graphe pondéré dont les sommets sont Q, l'alphabet A et les poids sont pris dans k. De plus celle de I (resp. T) correspond à la donnée de flêches rentrantes (resp. sortantes) marquées avec des poids. Dans tout ce processus, on peut ne pas indiquer les flêches de poids nul.
Ce type d'automates généralise les automates (booléens) de la théorie des langages (que l'on obtient alors pour $k = \mathbb{B}$) est une machine qui prend un mot en entrée et retourne un coefficient (dans k) en sortie. Son comportement est donc une fonction $\mathcal{A} : \ A^* \to k$ (que l'on peut noter, de manière équivalente, comme une série $\mathcal{A} = \sum_{w\in A^*} \mathcal{A}(w)w$).

Calcul du poids $\mathcal{A}(w)$. —
On étend d'abord la fonction de transition M à A^* par

$$M(\epsilon) = I_{Q\times Q},\ M(w) = M(a_1a_2\cdots a_n) = M(a_1)M(a_2\cdots M(a_n) \tag{14}$$

où $I_{Q\times Q}$ est la matrice identité de format $Q \times Q$. Le calcul du poids d'un mot est alors, par définition,

$$\mathcal{A}(w) := IM(w)T \tag{15}$$

d'après la règle de multiplication des matrices, on a bien que $IM(w)T$ est une matrice de format 1×1 et donc un élément de k. Le lien avec le graphe de l'automate est donné par la proposition suivante :

Proposition 4.1 *Soit, pour deux états r,s et un mot $w \in A^*$*

$$\mathcal{A}^{r,s}(w) := I_r \left(\sum_{\substack{c,\ \text{chemin}\ \ l(c)=w \\ t(c)=r,\ \ h(c)=s}} weight(c) \right) T_s \tag{16}$$

alors

$$\mathcal{A}(w) = \sum_{r,s\in Q} \mathcal{A}^{r,s}(w) \tag{17}$$

Cette proposition a le sens intuitif suivant :

1. l'équation (16) donne le poids calculé comme au paragraphe précédent
   - on fait le bilan parallèle (c'est à dire une somme) des poids des chemins qui joingnent r à s
2. on multiplie (à gauche si c'est non commutatif) par le poids d'entrée en r
3. on multiplie (à droite si c'est non commutatif) par le poids de sortie en s

4.2.3 Premiers automates

1. Longueur totale $\sum_{w\in A^*} |w|w$
2. Comptage des a, $\sum_{w\in A^*} |w|_a w$ et des b, $\sum_{w\in A^*} |w|_b w$
3. Produit des degrés partiels $\sum_{w\in A^*} |w|_a|w|_b w$
4. Autres produits $\sum_{w\in A^*} F_{|w|}|w|w$, $\sum_{w\in A^*} F_{|w|_a}|w|_b w$

Fin du tronc commun

4.2.4 Composition des automates

Somme et multiplication par un coefficient constant

Produit de Hadamard

Produit (de concaténation)

Nous avons vu que nous pouvions coder de "l'infini dans du fini" en considérant les suites ultimement périodiques que sont les développements illimités des rationnels. Nous allons

voir qu'il en est de même pour la production des automates finis, en effet, un automate fini, dès qu'il possède un chemin réussi qui comporte un boucle, reconnaît un langage infini.

Exercice 4.2 *Montrer que cette condition est suffisante, autrement dit, si aucun chemin réussi ne comporte de boucle, alors le langage reconnu par l'automate est fini.*

Commençons par un exemple : On considère un automate (booléen), d'ensemble d'états Q et dont les transitions sont étiquetées par un alphabet A. Cet automate, via la correspondance (graphes $\leftrightarrow$ matrices) peut être vu comme un triplet (I,T,M) avec :

$$\begin{cases} \bullet \text{ Un vecteur d'entrée } I \in k^{1\times Q} \\ \bullet \text{ Une famille de matrices de transition } M : A \to k^{Q\times Q} \\ \bullet \text{ Un vecteur de sortie } T \in k^{Q\times 1} \end{cases}$$

Dans les automates usuels, les scalaires sont pris dans $\{0,1\}$. Si on considère ces nombres comme des entiers naturels, l'opération $w \to IM(w)T$ donne le nombre de chemins réussis. Une expression rationnelle du comportement de l'automate (tenant compte des multiplicités) résulte du calcul suivant

$$\sum_{w\in\Sigma^*} (IM(w)T)w = I(\sum_{w\in\Sigma^*} M(w)w)T = I(Id_n - \sum_{a\in\Sigma} M(a)a)^{-1}T$$

si on note $M_\Sigma = \sum_{a\in\Sigma} M(a)a$, on a $M_\Sigma^* = (Id_n - \sum_{a\in\Sigma} M(a)a)^{-1}$. C'est la matrice dont l'entrée d'adresse (i,j) est la somme

$$\sum_{\substack{w \text{ étiquette} \\ \text{un chemin de } i \text{ vers } j}} (\text{nb de chemins } i \to j \text{ d'étiquette } w)w$$

par exemple la matrice

$$M_\Sigma = \begin{pmatrix} a & a \\ b & 0 \end{pmatrix}$$

a pour étoile

$$M_\Sigma^* = \begin{pmatrix} (a+ab)^* & (a+ab)^*a \\ b(a+ab)^* & (ba^*a)^* \end{pmatrix}$$

il est facile de voir que les séries associées sont sans multiplicité (i.e. pour (i,j) et w donnés il existe au plus un chemin d'étiquette w), mais ce n'est pas le cas pour

$$Q_\Sigma = \begin{pmatrix} a & a \\ b & a \end{pmatrix}$$

qui a pour étoile

$$Q_\Sigma^* = (\ (a+aa^*b)^* \quad (a+aa^*b)a^*aa^*b(a+aa^*b)^* \quad (a+ba^*a)^*\ )$$

Exercice 4.3 *1) Dessiner les structures de transition (c'est à dire les automates sans vecteurs d'entrée et sortie) associés aux matrices M_Σ,Q_Σ.*
2) a) Montrer, en utilisant un raisonnement sur les chemins dans un graphe étiqueté convenable, que pour deux lettres, on a $(a+b)^* = (a^*b)a^*$ (élimination de Lazard monoïdale).
b) Appliquer cette identité pour trouver une autre forme de $(a+aa^*b)^*$.

c) Montrer que $a^*aa^* = a\frac{1}{(1-a)^2} = \sum_{n\geq 1} na^n$.
d) Si un mot ne se termine pas par b, sa multiplicité dans $(a^*aa^*b)^*$ *est nulle, mais s'il s'écrit* $w = a^{n_1}ba^{n_2}b\cdots a^{n_k}b$, *on a* $(w,(a^*aa^*b)^*) = n_1 + n_2 + \cdots n_k$. *En déduire le développement (i.e. les multiplicités des mots) de* $(a^*aa^*b)^*a^*$, *puis des 4 coefficients de la matrice* Q_Σ^*.
3) a) Soit l'alphabet à quatre lettres $\Sigma = \{a_{11},a_{12},a_{21},a_{22}\}$, *montrer directement en raisonnant sur les chemins, que si* $G = \begin{pmatrix} a_{11} & a_{12}a_{21} \\ & a_{22} \end{pmatrix}$ *on a*

$$G^* = \begin{pmatrix} A_{11} & A_{11}a_{12}a_{22}^* \\ a_{22}^*a_{21}A_{11} & A_{22} \end{pmatrix}$$

avec

$$A_{11} = (a_{11} + a_{12}a_{22}^*a_{21})^*,\ A_{22} = (a_{22} + a_{21}a_{11}^*a_{12})^*$$

b) Expliquer en quoi ces formules fournissent un algorithme permettant de calculer l'étoile de toute matrice de séries propres.

Exemple 4.4 *Soit* L_n *le langage fini formé des mots* w *tels que* $|w|_a + 2|w|_b = n$.
a) Écrire les premiers termes $L_0,L_1,L_2\cdots$.
b) Calculer $|L_n|$ *à l'aide d'une récurrence simple.*
c) Montrer que $SG = \sum_n |L_n|t^n = (t+t^2)^* = \frac{1}{1-t-t^2}$.
d) Faire le lien avec le nombre de pavages d'un rectangle $2\times n$ *par des dominos* 2×1 *([?] pp 321) comment coder les pavages, les énumérer, les générer.*
e) À l'aide des décalages, former l'automate qui reconnait la série S.

On a un analogue parfait de ce qui se passe pour les rationnels positifs. Plus précisément :

Exercice 4.5 *A) On considère les arbres* $1-2$ *qui sont les arbres à* 1 *ou deux fils.*
À chaque arbre $1-2$, *dont les noeuds internes sont signés par "+" s'ils ont deux fils et "*$()^{-1}$*" s'ils en ont un on fait correspondre une fraction (i.e. son évaluation avec les feuilles en* 1*) donnée par la règle récursive*

$$ev(\bullet) = 1;\ ev((\mathcal{A}_1,\mathcal{A}_2)) = ev((\mathcal{A}_1) + ev(\mathcal{A}_2);\ ev((\mathcal{A})) = \frac{1}{ev(\mathcal{A})}$$

montrer que l'ensemble des valeurs obtenues est $\mathbb{Q}_+^*$. *Est-ce que la représentation est unique ? Est-ce qu'elle englobe les fractions continues ? Comment caractériser les arbres qui les donnent ?*
B) On considère les séries sur un alphabet A *(i.e. fonctions* $A^* \to k$ *où* k *est un semi-anneau (i.e. suffisant pour faire la calcul matriciel).*
a) Montrer que les conditions suivantes sont équivalentes :
i) La série S *est l'évaluation d'une expression rationnelle.*
ii) La série S *est combinaison linéaire d'un ensemble de séries* $S_1,S_2,\cdots S_n$ *qui est (linéairement) stable par décalages soit*

$$(\forall x \in A)(\forall i \in [1..n])(x^{-1}S_i = \sum_{0\leq j\leq n} \mu_{i,j}(x)S_j$$

iii) Il existe $\lambda \in K^{1\times n}$, $\mu : A \to K^{n\times n}$,$\gamma \in K^{1\times n}$ *tels que pour tout* $w \in A^*$, $(S,w) = \lambda\mu(w)\gamma$ *(où* $\mu()$ *dénote encore l'extention de* μ *à* A^**).*

Lorsque l'on a une partie $X \in A^*$, on peut se demander :
Quel est le langage $L(X)$ engendré par X ?
c'est à dire les suites finies d'instructions (i.e. le sous-monoïde engendré). On a $L(X) = \sum_{n\geq 0} X^n$ à coefficients dans $\mathbb{B}$. La même somme à coefficients dans $\mathbb{N}$ contient plus d'informations (soit le nombre de façons d'obtenir w comme produit de facteurs dans X).
Automates. Automates à multiplicité (notion de coût). Comportement d'un automate. Séries (exemples), rationnelles.
Passage SGO $\leftrightarrow$ Aut $\leftrightarrow$ Exp. rat.
Exemples de $\mathbb{N}$ et $\mathbb{Z}$ automates.
Séries génératrices (rationnelles -arbres de Fibonacci- et non rationnelles -arbres binaires, chemins de Dyck-). Résolution des premières récurrences, décalage et Δ. Complexité du comptage des boucles. Arbres $1-2$.

Références

[1] Cohen H., *A Course in Computational Algebraic Number Theory*. Springer (1993)

[2] Char B.W., Geddes K.O., Gonnet G.H., ali.., *Maple V Library Reference Manual*, Springer (1992).

[3] Char B.W., Geddes K.O., Gonnet G.H., ali.., *Maple V Language Reference Manual*, Springer (1992).

[4] Davenport J., Siret Y., Tournier E., *Calcul formel*, Masson (1986)

[5] Demazure M., Cours d'algèbre : Divisiilité, Primalité, codes. Cassini (1997).

[6] von zur Gathen J. and Gerahrd J. Modern Computer Algebra. Cambridge (1999).

[7] Knuth D., *The art of computer programming* Tome I. Addison-Wesley (1981)

[8] Knuth D., *The art of computer programming* Tome II. Addison-Wesley (1981)

[9] Naudin P., Quitté C., *Algorithmique Algébrique* Masson (1992)