Notes de cours sur la théorie des fonctions régulières de coût

Thomas Colcombet

Modélisation par automates finis
MPRI 2011/2012

Ces notes sont en cours de rédaction. Elles contiennent certaines erreurs et typos.

Table des matières

Introduction

Cette partie de cours décrit la notion de fonction régulière de coût, une extension quantitative de la notion de langages réguliers de mots. Le principe de cette théorie est d'étudier non pas des langages, c.a.d. des applications des mots dans $\{\text{dedans}, \text{dehors}\}$, mais des fonctions dans $\omega + 1$ (les entiers augmentés de l'infini, noté ω). Ces fonctions sont étudiées modulo une relation d'équivalence $\approx$ qui élimine toute information concernant les valeurs exactes des fonctions, mais préserve suffisamment d'information pour tester des propriétés telles que «est-ce que la fonction est bornée ?».

Cette théorie est la continuation de travaux dont l'objectif était la résolution de problèmes précis de décision en théorie des automates et en logique. Citons :

La hauteur d'étoile. Entrée : un language régulier L et un entier k. Sortie : oui si le langage L peut-être décrit au moyen d'une expression rationnelle de hauteur k, non sinon. (problème posé par Eggan en 63 [7]).

La puissance finie. Entrée : un langage régulier L. Sortie : oui s'il existe n tel que $L^n = L^*$, non sinon (problème du à Brzozowski).

La substitution finie. Entrée : deux langages K, L. Sortie : oui s'il existe une substitution σ envoyant chaque lettre sur un langage fini, et telle que $\sigma(L) = K$, non sinon.

La borne sur point-fixe. Dans ce problème, qui nécessiterait plus de formalisme pour être introduit formellement, il s'agit de décider si le point fixe d'une formule logique peut-être atteint en un nombre borné d'itération sur une classe de structure. Ce problème, étudié en théorie des modèles, se décline en plusieurs modalités suivant le choix de la logique et la classe de structure. Il s'agit ici de la logique monadique du second-ordre, sur la classe des mots.

Tous ces problèmes ont été résolus [17,6,10,9,11,12,14,18,8,1,13,2] en utilisant une réduction à des questions de la forme «décider si la fonction F est bornée», où F est une fonction décrite finiment (par exemple au moyen d'automates ou d'expressions régulières possédant des capacités de comptage), et calculée en fonctions des entrées du problème. Ce cours a pour but la description d'une théorie, parallèle à la théorie des langages réguliers, et fournissant des outils systématiques pour la résolution de telles questions d'existence de bornes.

Notations

On utilise les notations suivantes : $\mathbb{A}$ est un alphabet (fini), $\mathbb{A}^*$ est l'ensemble des mots sur cet alphabet, $\mathbb{A}^+$ est l'ensemble des mots d'au moins une lettre. ε dénote le mot vide. Pour a une lettre fixée, et u un mot, $|u|_a$ représente le nombre d'occurrences de la lettre a dans le mot u.

L'ensemble $\omega+1$ est $\{0, 1, \ldots\} \cup \{\omega\}$, c.a.d, les entiers augmentés d'un «plus grand élément infini» ω. Tous les calculs de maximum et de minimum sont effectués dans cet ensemble. On prend donc comme conventions que $\min \emptyset = \omega$ et $\max \emptyset = 0$.

1 Définition des B-automates

Cette première section a pour but d'introduire le modèle des B-automates. Les B-automates sont des automates finis qui à chaque mot associent une valeur dans $\omega + 1$. Ils utilisent des compteurs qu'ils peuvent incrémenter, remettre à zéro, et observer. La fonction qu'ils calculent est le minimum sur toutes les exécutions de la plus grande valeur d'un compteur observée.

1.1 Coût d'une séquence

Nous commençons la description des automates en décrivant plus précisément comment calculer le résultat d'une série d'actions sur un compteur. On utilise trois *actions élémentaires* : incrément (`i`), remise à 0 (`r`=reset), et test (`c`='check'). Considérons une séquence $u \in \{\mathtt{i}, \mathtt{c}, \mathtt{r}\}^*$, et un compteur pouvant prendre des valeurs entières. La séquence est exécutée de gauche à droite. Au début le compteur a la valeur 0, puis il évolue à chaque lettre rencontrée, de la façon suivante :

– si la lettre est `i`, la valeur du compteur est augmentée de 1,
– si la lettre est `r`, la valeur du compteur est remise à 0,
– si la lettre est `c`, la valeur du compteur ne change pas, mais est *observée*.

On définit $C(u)$ comme l'ensemble des valeurs observées, et on définit le *coût* de la séquence comme $coût_B(u) = \max C(u)$ (rappelons que $\max \emptyset = 0$ par convention).

Une manière plus directe de calculer le coût d'une séquence est par exemple d'utiliser le fait suivant.

Fait 1 $coût_B(u) = \max\{|v|_i \ : \ u = u'v\mathtt{c}u'',\ |v|_{\mathtt{r}} = 0\}$.

Soit Γ un ensemble fini de *compteurs*, l'ensemble des *actions (simples)* sur Γ est $A_\Gamma = \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^\Gamma$. Étant donné un mot u sur l'alphabet A_Γ, on définit $C(u)$ comme l'union des $C(u^\gamma)$ où u^γ est le mot obtenu en projetant chaque lettre de u sur sa composante $\gamma \in \Gamma$. On définit $coût_B(u)$ comme $\max C(u)$.

Cela s'obtient de manière identique en utilisant le fait suivant.

Fait 2 $coût_B(u) = \max\{coût(u^\gamma) \ : \ \gamma \in \Gamma\}$.

1.2 *B*-automates

Un *B-automate* $\mathcal{A} = \langle Q, \mathbb{A}, I, F, \Gamma, \Delta\rangle$ est décrit par :
– un ensemble fini d'*états* Q,
– un *alphabet* fini $\mathbb{A}$,
– un ensemble d'états *initiaux* $I \subseteq Q$,
– un ensemble d'états *finaux* $F \subseteq Q$,
– un ensemble de transitions $\Delta \subseteq Q \times \mathbb{A} \times A_\Gamma \times Q$.

Une *exécution* σ de $\mathcal{A}$ sur $u \in \mathbb{A}^*$ est une séquence $\sigma = (p_0, a_1, t_1, p_1)\dots(p_{n-1}, a_n, t_n, p_n)$ telle que :
– $(p_{i-1}, a_i, t_i, p_i) \in \Delta$,
– $a_1 \dots a_n = u$,
– $p_0 \in I$,
– $p_n \in F$.

Son coût est $coût(\sigma) = coût(t_1 \dots t_n)$.

On définit la valeur d'un mot $u \in \mathbb{A}^*$ pour $\mathcal{A}$ comme :

$$[\![\mathcal{A}]\!]_B(u) = \min\{coût(\sigma) \ : \ \sigma \text{ exécution de } \mathcal{A} \text{ sur } u\} \qquad \text{(calculé dans } \omega + 1)$$

(rappelons que $\min \emptyset = \omega$, par convention)

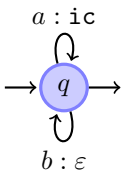
Remark 1. S'il n'existe aucune exécution sur u, alors $[\![\mathcal{A}]\!]_B = \omega$.

Remark 2. Si $\mathcal{A}$ ne possède aucun compteur (i.e., $\Gamma = \emptyset$) alors $\mathcal{A}$ peut être vu comme un automate non-déterministe acceptant le langage régulier $L = \mathcal{L}(\mathcal{A})$. On a alors pour tout mot u :

$$[\![\mathcal{A}]\!](u) = \chi_L(u) = \begin{cases} 0 & \text{si } u \in L, \\ \omega & \text{sinon.} \end{cases}$$

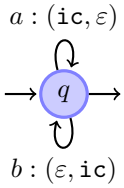
En particulier, si $\mathcal{A}$ et $\mathcal{A}'$ sont des B-automates sans compteurs, alors $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ ssi $[\![\mathcal{A}]\!] \geq [\![\mathcal{A}']\!]$.

Exemple 1 *Le B-automate suivant compte le nombre d'occurrences de la lettre a.*



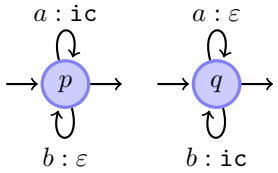
On utilise les conventions suivantes. Les états sont représentés par des cercles. Les états initiaux ont une flèche entrante sans origine, et les états finaux sont origines d'une flèche sans destination. Une transition (p, a, t, q) est représentée comme une flèche d'origine p, de destination q, étiquetée par $a : t$. Dans cet exemple, comme il n'y a qu'un compteur, t est représenté comme l'action sur l'unique compteur.

Exemple 2 *Le B-automate suivant calcule le maximum du nombre d'occurrences de la lettre a et du nombre d'occurrences de la lettre b. Il utilise deux compteurs.*



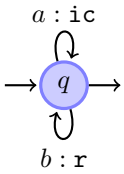
Cette fois-ci, on utilise deux compteurs, et les actions sont donc représentées comme une paire formée de l'action sur le premier compteur, suivie de l'action sur le deuxième compteur.

Exemple 3 *Le B-automate à un compteur suivant calcule le minimum du nombre d'occurrences de la lettre a et du nombre d'occurrences de la lettre b.*

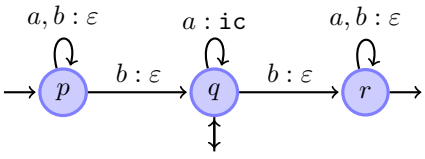


Cette fois-ci, l'automate doit deviner, en utilisant le non-déterminisme, s'il compte le nombre d'occurrences de a ou le nombre d'occurrences de b.

Exemple 4 *Le B-automate suivant calcule la longueur maximale d'un segment formé uniquement de a.*



Exemple 5 *Le B-automate suivant calcule la longueur minimale d'un segment formé uniquement de a (un segment étant délimité ici, par, à gauche, soit le début du mot, soit une lettre b, et à droite, soit la fin du mot, soit une lettre b).*



On utilise l'abréviation $a, b : t$ annotant une flèche d'origine p et de destination q, pour dénoter qu'il existe les deux transitions (p, a, t, q) et (p, b, t, q).

Cet automate fonctionne en devinant, en faisant usage de son non-déterminisme, le segment de a qu'il s'agit de mesurer.

Exercice 6 *Déterminer ce que calculent les B-automates à deux compteurs suivant :*

Exercice 7 *Montrer qu'il est impossible de déterminiser les B-automates. Idée : prendre le deuxième automate de l'exemple précédent, et remarquer que les automates déterministes sur une lettre ont un comportement «quasi linéaire», c.a.d : si $[\![\mathcal{A}]\!]_B(a^n)$ tend vers l'infini, alors $[\![\mathcal{A}]\!]_B(a^{kn}) \geq n$ pour un certain entier k.*

1.3 Résultats élémentaires

Dans cette section, on établit certains résultats de clôture sur les fonctions calculées par B-automates.

Proposition 1. *Le support d'une fonction définie par B-automate est régulière (i.e., l'ensemble des mots de valeur finie).*

Idée de preuve : enlever les actions sur les compteurs pour obtenir un automate non-déterministe acceptant un langage.

Proposition 2. *L'ensemble des mots de valeur n est régulier.*

Idée : il est possible, au moyen d'un automate fini, de compteurs «jusqu'à n». Un automate fini peut donc simuler un B-automate pour toutes valeurs des compteurs inférieures à n.

Proposition 3. *Les fonctions calculées par B-automate sont closes sous* min.

Idée : effectuer l'union disjointe des deux automates. À titre d'exemple, on peut observer le rapport entre l'automate de l'exemple 1 et l'automate de l'exemple 3.

Proposition 4. *Les fonctions calculées par B-automate sont closes sous* max.

Idée : il s'agit cette fois-ci de construire un automate produit travaillant sur l'union disjointe des compteurs. À titre d'exemple, on peut observer le rapport entre l'automate de l'exemple 1 et l'automate de l'exemple 2.

Proposition 5. *Les fonctions calculées par B-automate sont closes sous* min*-projection (soit f un morphisme lettre à lettre, et $\mathcal{A}$ un B-automate, construire $\mathcal{A}'$ tel que $\mathcal{A}'(u) = \min\{\mathcal{A}(v) \ : \ f(v) = u\}$).*

Idée : remplacer toutes chaque transition (p, a, t, q) par une transition $(p, f(a), t, q)$.

2 Équivalence «de bornes»

Il est possible, comme conséquence d'un résultat de Krob [15], d'obtenir le résultat d'indécidabilité suivant :

Théorème 3 *Le problème de l'équivalence des fonctions calculées par B-automates est indécidable.*

Comme mentionné dans l'introduction, l'objectif de cette théorie n'est pas de calculer précisément les valeurs produites par les B-automates, mais plutôt de décider de questions comme : «est-ce que le fonction calculée par un B-automate est bornée ?»

Pour cette raison, on s'autorise à étudier les fonctions calculées par B-automates modulo une relation d'équivalence qui préserve les propriétés du type «existence de borne», mais élimine suffisamment d'information pour contourner les résultats d'indécidabilité comme celui du théorème 3. Le propos de cette section est d'introduire une relation d'équivalence permettant d'atteindre cet objectif.

2.1 Définition

Soit E un ensemble, et f, g deux applications de E dans $\omega + 1$, alors f est *dominée* par g, noté $f \preccurlyeq g$ si pour tout sous-ensemble $X \subseteq E$, si g est bornée sur X alors f est aussi bornée sur X. La notation $f \approx g$ signifie que, à la fois, $f \preccurlyeq g$ et $g \preccurlyeq f$. On appelle *fonction de coût* (sur E) une classe d'équivalence pour $\approx$.

Exemple 8 $|\cdot|_a \preccurlyeq |\cdot|$, *en effet, sur tout sous-ensemble de mots de longueur bornée, le nombre d'occurrence de a est aussi borné.*
En revanche $|\cdot|_a \not\preccurlyeq |\cdot|_b$, car sur l'ensemble $X = a^*$, $|\cdot|_b$ est bornée, alors que $|\cdot|_a$ ne l'est pas.
On peut aussi remarquer que si $f \leq g$ alors $f \preccurlyeq g$.

Pour montrer que des fonctions sont comparables pour $\preccurlyeq$, il est utile d'utiliser un raffinement de $\preccurlyeq$. On appelle *fonction de correction* ou *fonction d'étirement* une fonction croissante α de ω dans ω que l'on étend par $\alpha(\omega) = \omega$. On pose alors :

$$f \preccurlyeq_\alpha g \qquad \text{si} \qquad f \leq \alpha \circ g \ ,$$

et $f \approx_\alpha g$ si $f \preccurlyeq_\alpha g$ et $g \preccurlyeq_\alpha f$. L'idée est que $f \preccurlyeq_\alpha g$ signifie que, une fois étirée par α, g est supérieure à f.

Remarquons que la relation $\preccurlyeq_\alpha$ n'est pas transitive. En revanche, elle satisfait les propriétés suivantes :

Fait 4 *Si $f \preccurlyeq_\alpha g$ alors $f \preccurlyeq_{\alpha'} g$ pour tout $\alpha' \geq \alpha$.*
Si $f \preccurlyeq_\alpha g \preccurlyeq_{\alpha'} h$ alors $f \preccurlyeq_{\alpha' \circ \alpha} h$.

Exemple 9 *On a $|\cdot|_a + |\cdot|_b \approx_{\times 2} \max(|\cdot|_a, |\cdot|_b)$, en effet :*

$$\max(|\cdot|_a, |\cdot|_b) \leq |\cdot|_a + |\cdot|_b \leq 2\max(|\cdot|_a, |\cdot|_b) \ .$$

Proposition 6. *Les propriétés suivantes sont équivalentes :*

1. $f \preccurlyeq g$,
2. $\forall N. \exists M. \forall x \in E.\ g(x) \leq N \to f(x) \leq M$.
3. *il existe α tel que $f \preccurlyeq_\alpha g$.*

Remark 3. Pour tout langage L, $\chi_L \preccurlyeq \chi_K$ ssi $K \subseteq L$.

Example 1. Sur $E = N$, $(n \mapsto n) \approx (n \mapsto 2^n) \succ (\frac{n+(-1)^n n}{2}) \succ 0$.
Sur $E = N^2$, $(\max) \approx_{\times 2} (+)$
Sur $E = \{\text{ensemble des arbres finis}\}$, *taille* $\approx_{n \mapsto n^n} \max(degmax, hauteur)$
Sur $E = \{a, b\}^*$, $|\cdot| = |\cdot|_a + |\cdot|_b \approx \max(|\cdot|_a + |\cdot|_b) \succ |a| \succ 0$.

Fait 5 *Si $f \approx f'$ et $g \approx g'$ alors $\min(f, g) \approx \min(f', g')$ et $\max(f, g) \approx \max(f', g')$. Si $f_i \approx_\alpha g_i$ pour tout $i \in I$, alors $\inf_{i \in I} f_I \approx_\alpha \inf_{i \in I} g_I$. Idem pour* sup.

Fait 6 *L'ensemble des fonctions de E dans $\omega + 1$ quotienté par $\approx$ est un treilli.*
Il possède une plus petite classe : la classe des fonctions bornées.
Il posséde une plus grande classe : la classe de la fonction constante égale à ω.
Si E est dénombrable, le 'sous-treillis' induit par les fonctions de E dans ω possède un élément maximal (la classe de une/toutes les bijections de E sur ω).

Proposition 7. *L'ensemble des fonctions de E infini dénombrable dans ω possède un cardinal du continu de classes d'équivalences.*

Démonstration. On pose, sans perte de généralité, que E est l'ensemble des applications de $\mathbb{N}$ dans $\mathbb{N}$ nulles presque partout. Il y en a un nombre dénombrable. Pour tout $I \subseteq \mathbb{N}$ et $g \in E$, on pose $f_I(g) = \max\{g(i) \ : \ i \in I\}$.

Soient $I \neq J$. On suppose, sans perte de généralité, qu'il existe $i \in I \setminus J$. Considérons la fonction g_n qui est nulle partout sauf pour $g_n(i) = n$. On a alors $f_I(g_n) = n$, et $f_J(g_n) = 0$. Ainsi, f_J est bornée sur $Y = \{g_n \ : \ n \in \mathbb{N}\}$, alors que f_I ne l'est pas. On en déduit $f_I \not\approx f_J$.

Comme l'ensemble des parties de $\mathbb{N}$ a le cardinal du continu, on en déduit qu'il existe autant de classes d'équivalence pour la relation $\approx$.

3 B-automates modulo $\approx$

3.1 Premières remarques

Example 2. $|\cdot|_a + |\cdot|_b = \max(|\cdot|_a + |\cdot|_b)$ On a vu que ces deux fonctions sont acceptées par des B-automates. Ceux-ci sont donc équivalent modulo $\approx$.

La proposition suivante montre que, même considéré modulo $\approx$, les B-automates ne peuvent être déterminisés.

Proposition 8. *La fonction $a^n b^m \mapsto \min(m, n)$ ne peut être reconnue par un B-automate déterministe modulo $\approx$.*

Démonstration. Par la contraposée, supposons qu'il existe une fonction de correction α et un B-automate $\mathcal{A}$ qui définisse une fonction $\approx_\alpha$-équivalente à $a^n b^m \mapsto \min(m, n)$. Notons $\delta(u)$ l'état atteint à partir de l'état initial en lisant le mot u en entrée.

On commence par analyser la structure de l'automate. Remarquons tout d'abord que le mot $a^m b^n$ n'a pas la valeur ω, et donc $\delta(a^m b^n)$ existe pour tous m, n, et est final. Comme l'ensemble d'états est fini, il existe k, l tels que $\delta(a^k) = \delta(a^{k+l})$. De même, il existe k', l', tels que $\delta(a^k b^{k'}) = \delta(a^k b^{k'+l'})$.

Supposons qu'il existe un compteur γ tel que sur la boucle parcourue en lisant a^l a partir de l'état $\delta(a^k)$, γ est incrémenté, mais pas remis à 0. Alors, on en déduit que $[\![\mathcal{A}]\!](a^{k+nl}) \geq n$, et donc $[\![\mathcal{A}]\!]$ n'est pas borné sur a^{k+nl}, alors que $a^n b^m \mapsto \min(m, n)$ l'est. Contradiction. De même, il est impossible qu'un compteur soit incrémenté mais non remis à 0 lorsque l'on parcours la boucle d'origine $\delta(a^k b^{k'})$ obtenue par lecture du mot $b^{l'}$. Ainsi, on sait que tous les compteurs sont soit remis à 0, soit jamais incrémentés dans ces boucles. On en déduit que $[\![\mathcal{A}]\!](a^{k+nl} b^{k'+nl'}) \leq k + l + k' + l'$, et donc $[\![\mathcal{A}]\!]$ est borné sur $\{a^{k+nl} b^{k'+nl'} \ : \ n \in N\}$, alors que $a^n b^m \mapsto \min(m, n)$ ne l'est pas. On aboutit de nouveau à une contradiction.

3.2 Équivalence avec les automates avec ϵ-transition

Un *B-automate avec ϵ-transition* est un B-automate sur l'alphabet $A \cup \{\epsilon\}$. Étant donné $u \in A \cup \{\epsilon\}$, $\overline{u}^\epsilon$ est le mot de A^* obtenu de u en éliminant toutes les occurrences de la lettre ϵ. On pose pour tout $u \in A^*$:

$$[\![\mathcal{A}]\!]^\epsilon(u) = \min\{[\![\mathcal{A}]\!](v) \ : \ \overline{v}^\epsilon = u\} \ .$$

On remarque que dans le cas sans compteurs, cette définition est consistante avec la notion habituelle d'automates avec ϵ-transitions.

L'objectif est d'établir la proposition suivante.

Proposition 9. *Les B-automates avec ϵ-transitions sont $\approx$-équivalents aux B-automates (sans ϵ-transitions).*

On définit l'opération de produit $\cdot$ sur $\varepsilon, \mathtt{ic}, \mathtt{r}$ comme le maximum pour l'ordre $\varepsilon < \mathtt{ic} < \mathtt{r}$. Ce produit est étendu composante par composante à $A_\Gamma = \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^\Gamma$. Soit $\pi : \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^* \to \{\varepsilon, \mathtt{ic}, \mathtt{r}\}$ l'extension du produit à un mot de longueur quelconque $u \in A_\Gamma^*$ (avec $\pi(\varepsilon) = \varepsilon$).

Lemme 7 *Pour tout $u_1, \ldots, u_n \in A_\Gamma^*$, $coût(\pi(u_1) \ldots \pi(u_n)) \leq coût(u)$, où $u = u_1 \ldots u_n$.*
Si $|u_i| \leq k$ pour tout $i = 1, \ldots, n$, alors $coût(u) \leq \alpha(coût(\pi(u_1) \ldots \pi(u_n)))$ où $\alpha(x) = k(x + 2)$.

Démonstration. Soit $N = \mathit{coût}(\pi(u_1)\dots\pi(u_n))$. Par définition de *coût*, il existe un facteur $u_i \dots u_j$ de u et un compteur $\gamma \in \Gamma$ tels que γ n'est jamais remis à 0 dans $\pi(u_i)\dots\pi(u_j)$ et est incrémenté N fois dans $\pi(u_i)\dots\pi(u_j)$. D'après la définition du produit, on en déduit que γ n'est jamais remis à 0 dans $u_i \dots u_j$, et qu'il existe N indices $l \in \{i, \dots, j\}$ tel que γ est incrémenté dans u_l. On en déduit que $\mathit{coût}(u_i \dots u_j) \geq N$, et par conséquent $\mathit{coût}(u) \geq N$.

Seconde implication. Soit $N = \mathit{coût}(u)$. Il existe un compteur γ et un facteur v de u tel que γ n'est jamais remis à 0 dans v, et est incrémenté N fois. Si v est le facteur d'un des u_i, alors il est est de longueur au plus k, et on obtient directement $\mathit{coût}(u) = N \leq |v| \leq k \leq \alpha(\mathit{coût}(\pi(u_1)\dots\pi(u_n))$. Sinon, v se décompose en $v'u_i \dots u_j v''$, ou v' est un suffixe de u_{i-1} (ou vide), et v'' est un préfixe de u_{j+1} (ou vide). Comme $|v'| \leq |u_{i-1}| \leq k$, et $|v''| \leq |u_{j+1}| \leq k$, on en déduit que γ est incrémenté au moins $N - 2k$ fois dans $u_i \dots u_j$. Soit I l'ensemble des $l = 1 \dots j$ tels que $\pi(u_l) = \texttt{ic}$. Tout incrément de $u_i \dots u_j$ a lieu dans un u_l pour $l \in I$, de plus chaque u_l pour $l \in I$ contient au plus k incréments de γ. On en déduit que $N - 2k \leq k|I|$, et par conséquent $\mathit{coût}(u) \leq k(|I| + 2) \leq \alpha(\mathit{coût}(\pi(u_1)\dots\pi(u_n)))$ avec $\alpha(x) = k(x + 2)$. □

Construction: On part de l'automate $\mathcal{A} = \langle Q, A, I, F, \Gamma, \Delta\rangle$ avec ϵ-transitions. Soit $\sigma = (p_0, a_1, t_1, p_1)\dots(p_{n-1}, a_n, t_n, p_n)$ un chemin dans $\mathcal{A}$, notons $\pi_1(\sigma)$ le mot $a_1 \dots a_n$, et $\pi_2(\sigma)$ la valeur $\pi(t_1 \dots t_n)$. Dans ce cas σ est appelé $a_1 \dots a_n$-chemin.

On construit l'automate $\mathcal{A}' = \langle Q, A, I, F, \Gamma, \Delta'\rangle$ où Δ' contient une transition (p, a, t, q) ssi il existe un chemin σ' de p à q dans $\mathcal{A}'$ tel que $\pi_1(\sigma') = a$ et $\pi_2(\sigma') = t$ (l'exécution σ' est appelée un *témoin* de (p, a, t, q)). Si $\mathcal{A}(\varepsilon) < \omega$, on ajoute à $\mathcal{A}'$ un état isolé à la fois initial et final.

Lemme 8 *S'il existe un ε-chemin σ de p à q, alors il existe un ε-chemin σ' de p à q de longueur au plus $|A_\Gamma||Q| - 1$ tel que $\pi_2(\sigma) = \pi_2(\sigma')$.*

Toute transition (p, a, t, q) de $\mathcal{A}'$ admet un chemin témoin de longueur au plus $K = 2|A_\Gamma||Q| - 1$.

Démonstration. Il suffit de montrer que pour tout ε-chemin σ de p à q de longueur au moins $|A_\Gamma||Q|$ il existe un ε-chemin σ' de p à q de longueur strictement inférieure tel que $\pi_2(\sigma) = \pi_2(\sigma')$. ayant même état initial et final, et tel que $\pi_1(\sigma') = \varepsilon$ et $\pi_2(\sigma') = \pi_2(\sigma)$. Pour cela il suffit de remarquer que σ peut être décomposé en $\sigma_1\sigma_2\sigma_3$ tel que $|\sigma_2| \geq 1$, σ_1 et σ_2 ont même état final et $\pi_2(\sigma_1) = \pi_2(\sigma_1\sigma_2)$ (pour cela, on remarque que la paire $(\pi_2(\sigma_1), q)$ ne peut prendre que $|A_\Gamma||Q|$ valeurs, donc après $|A_\Gamma||Q|$, il y a au moins une répétition). On en déduit que $\sigma' = \sigma_1\sigma_3$ convient. En appliquant cet argument de raccourcissement par récurrence, on peut réduire la longueur de σ à au plus $|A_\Gamma||Q| - 1$.

Considérons maintenant un chemin témoin σ de (p, a, t, q), il peut s'écrire $\sigma_1\delta\sigma_2$ où σ_1 et σ_2 sont des ε-chemins et $\pi_1(\delta) = a$. Par l'argument précédent on construit σ'_1 à partir de σ_1 et σ'_2 à partir de σ_2. Soit $\sigma' = \sigma'_1\delta\sigma'_2$. Par construction σ' est témoin de (p, a, t, q), et la longueur de σ' est au plus $2(|A_\Gamma||Q| - 1) + 1$. □

Il est maintenant possible d'achever la preuve de la proposition 9 en établissant que pour tout $u \in A^*$, $[\![\mathcal{A}']\!](u) \leq \mathcal{A}(u)$, et $[\![\mathcal{A}]\!](u) \leq K([\![\mathcal{A}']\!](u) + 1)$ (ce qui implique, $[\![\mathcal{A}']\!] \approx [\![\mathcal{A}]\!]$).

Démonstration. Première inégalité. Si $[\![\mathcal{A}]\!](u) = \omega$, on obtient directement $[\![\mathcal{A}']\!](u) \leq [\![\mathcal{A}]\!](u)$. Sinon, on sépare deux cas. Si $u = \varepsilon$. Comme $[\![\mathcal{A}]\!](\varepsilon) < \omega$, il existe (par construction) un état initial et final dans $\mathcal{A}'$, et par conséquent $[\![\mathcal{A}']\!](\varepsilon) = 0$. On en déduit $[\![\mathcal{A}']\!](\varepsilon) \leq [\![\mathcal{A}']\!](\varepsilon)$. Sinon, $u = a_1 \dots a_n$ est non vide. Comme $[\![\mathcal{A}]\!](u) < \omega$, il existe une exécution σ de $[\![\mathcal{A}]\!]$ de coût $[\![\mathcal{A}]\!](u)$ sur le mot u. Cette exécution se décompose en $\sigma_1 \dots \sigma_n$, où $\pi_1(\sigma_i) = a_i$. Soit p_0 l'état initial de σ_1 et p_i l'état final de σ_i. L'exécution σ_i est un témoin de $\delta_i = (p_{i-1}, a_i, \pi_2(\sigma_i), p_i)$ pour tout $i = 1, \dots, n$. On en déduit que $\sigma' = \delta_1 \dots \delta_n$ est une exécution de $\mathcal{A}'$ sur u. Grâce au lemme 7, son coût est au plus $\mathit{coût}(\sigma) = [\![\mathcal{A}]\!](u)$.

Réciproquement. Soit u un mot. Si $u = \varepsilon$ et $[\![\mathcal{A}']\!](u) < \omega$, alors nécessairement $\mathcal{A}'$ contient un état à la fois initial et final. Soit cet état était déjà présent dans $\mathcal{A}$, auquel cas $[\![\mathcal{A}]\!](u) = 0$. Soit cet état est l'état que l'on a rajouté parce que $[\![\mathcal{A}]\!](\varepsilon) < \omega$. D'après le lemme 8, cela signifie que ε est accepté par un une exécution de longueur au plus $|A_\Gamma||Q| - 1$. On en déduit $[\![\mathcal{A}]\!](u) \leq K \leq K([\![\mathcal{A}']\!](u) + 1)$. Sinon $u \neq \varepsilon$. Soit σ' une exécution de $\mathcal{A}'$ sur $u = a_1 \dots a_n$. Cette exécution s'écrit $\sigma' = \delta'_1 \dots \delta'_n$. Grâce au lemme 8 il existe pour tout $i = 1, \dots, n$ un chemin σ_i témoin de δ'_i de longueur au plus K. Considérons le chemin $\sigma = \sigma_1 \dots \sigma_n$. Il s'agit par construction d'une exécution de $\mathcal{A}$ sur u. De plus, d'après le lemme 7, $\mathit{coût}(\sigma) \leq K\,\mathit{coût}(\sigma')$. □

3.3 Équivalence avec les B-automates hiérarchiques

Automates hiérarchiques Idée: On restreint l'utilisation possible des compteurs. Les compteurs $1, \ldots, k$ sont ordonnés, et incrémenter ou remettre à zéro un compteur réinitialise tous les compteurs inférieurs.

Définition: Un B-automate est dit *hiérarchique* si son ensemble de compteurs est de la forme $\Gamma = \{1, \ldots, k\}$, et les seules actions H_k sur les compteurs sont :

- R_l pour $l = 0, \ldots, k$, tel que $R_l(\gamma) = \begin{cases} \texttt{r} & \text{si } \gamma \leq l \\ \varepsilon & \text{sinon} \end{cases}$
- I_l pour $l = 1, \ldots, k$, tel que $I_l(\gamma) = \begin{cases} \texttt{r} & \text{si } \gamma < l \\ \texttt{ic} & \text{si } \gamma = l \\ \varepsilon & \text{sinon.} \end{cases}$

Remark 4. On peut utiliser R_0 même s'il n'existe aucun compteur de ce numéro. On peut aussi noter cette action tout simplement ε.

Remark 5. Pour le produit utilisé ci dessus, on obtient que $c \cdot c'$ pour c, c' dans H_k est le maximum dans l'ordre :

$$R_0 < I_1 < R_2 < \quad \ldots \quad < I_k < R_k \ .$$

On en déduit en utilisant la même construction que les ϵ-transitions peuvent-être éliminées des B-automates hiérarchiques.

Équivalence Théorème: Les B-automates hiérarchiques sont $\approx$-équivalents aux B-automates.

On utilise une technique issue de la théorie des automates sur les mots et les arbres infinis.

Étape 1: On construit un automate hiérarchique (déterministe) LAR_Γ sur l'alphabet A_Γ tel que :

$$LAR_\Gamma \approx \textit{coût} \ .$$

Construction: On pose $k = |\Gamma|$. LAR_Γ est défini par :

- les états sont des permutations de Γ, i.e., des bijections de $\{1, \ldots, k\}$ sur Γ,
- l'alphabet en entrée est A_Γ,
- les compteurs sont $\{1, \ldots, k\}$,
- depuis l'état π, et en lisant la lettre $t \in A_\Gamma = \{\varepsilon, \texttt{ic}, \texttt{r}\}^\Gamma$, on calcule :

$$i = \max\{l \ : \ t(\pi(l)) = \texttt{ic}\} \qquad \text{et} \qquad r = \max\{l \ : \ t(\pi(l)) = \texttt{r}\} \qquad (0 \text{ par défaut}).$$

on utilise la transition :
- (π, t, I_i, π') si $i > r$,
- (π, t, R_r, π') si $r \geq i$ (et comme un cas particulier $R_0 = \varepsilon$ si $i = r = 0$).

où π' est obtenu de π en mettant les compteurs γ tels que $t(\gamma) = r$ en tête de la permutation, et en laissant les autres ordres relatifs inchangés.

Lemme 9 *Considérons une transition (π, t, h, π') de LAR_Γ. Considérons les indices dans π et π' d'un compteur γ, alors :*

- *si γ n'est pas remis à zéro dans h, son indice ne peut qu'augmenter,*
- *si γ n'est pas remis à zéro dans h alors son indice augmente strictement si et seulement si un compteur d'indice supérieur est remis à 0.*

Démonstration. Par construction. Il suffit de montrer pour une lettre, puis par induction sur la longueur.

Lemme 10 *Si u est une séquence d'incréments hiérarchisés de longueur n^k, alors $\textit{coût}(u) \geq n$.*

Démonstration. Par récurrence sur k on prouve que toute séquence d'incréments hiérarchisés ne touchant que aux compteurs inférieurs ou égaux à k et de longueur au moins n^k à un coût au moins égal à n. Pour $k = 0$, la longueur de la séquence est nulle, et donc la propriété est vraie. Sinon, la séquence u s'écrit comme $u_0 I_k u_1 \dots I_k u_l$ où les u_i's ne contiennent pas d'incréments de k. Si l'un des u_i's a une taille au moins égale à n^{k-1}, on conclut par l'hypothèse de récurrence. Sinon, $n^k \leq |u| = |u_0|+|u_1|+\cdots+|u_l|+l < n^{k-1}(l+1)$. On en déduit que $n < l+1$, i.e., $l \geq n$. Le coût et de la séquence est donc bien au moins n.

Lemme 11 *Pour tout $u \in A_\Gamma^*$, $coût(u) \approx_\alpha LAR_\Gamma(u)$ où $\alpha(n) = k(n+1)^k - 1$.*

Démonstration. Fixons une exécution ρ de LAR_Γ (sur un mot $u \in A_\Gamma^*$). Soit $n = coût(\rho)$. Cela veut dire qu'il existe un facteur ρ' de ρ et $l \in \{1, \dots, k\}$ tel que I_l est effectué n fois, et que tous les autres I_m et R_m ont lieu pour $m < l$. Le compteur $\pi(m)$ est inchangé dans ρ', donc $\pi(m)$ est incrémenté n fois dans ρ'. D'où $coût(u) \geq n$. On obtient $LAR_\Gamma(u) \leq coût(u)$.

Réciproque : Supposons que $coût(u) \geq kn^k$, cela veut dire qu'il y a un facteur ρ' de ρ et un compteur γ tel que γ est incrémenté kn^k fois dans ρ', sans jamais être remis à 0. D'après le Lemme 9, l'indice de γ ne peut qu'augmenter dans ρ'. Cela signifie qu'il existe un facteur ρ'' de ρ' dans lequel γ conserve le même indice l en étant incrémenté n^k fois. Au cours de ρ' aucun $R_{l'}$ pour $l' \geq l$ n'est effectué, et au moins n^k incréments hiérarchisés $I_{l'}$ pour $l' \geq l$ sont effectués. Par le lemme 10, on obtient $coût(\rho') \geq n$, et par conséquent $LAR_\Gamma(u) \geq n$. On en déduit que :

$$coût(u) \preccurlyeq_\alpha LAR_\Gamma(u) \qquad \text{pour } \alpha(n) = k(n+1)^k - 1 \ .$$

Étape 2: On part d'un B-automate $\mathcal{A}$ (compteurs Γ) et on effectue le produit avec LAR_Γ pour obtenir un automate équivalent hiérarchique.

Construction: $\mathcal{A} = \langle Q, A, I, F, \Gamma, \Delta\rangle$ et $LAR_\Gamma = \langle P, A_\Gamma, H_k, \delta\rangle$

On construit $\mathcal{A}' = \langle Q \times P, A, I \times P, F \times P, H_k, \Delta'\rangle$ où :

$$\Delta' = \{((q,p), a, h, (q',p')) \ : \ (q,a,m,q') \in \Delta, \ (p, \mathbf{M}, h, p') \in \delta\} \ .$$

L'automate hiérarchique $\mathcal{A}'$ est $\approx$-équivalent à $\mathcal{A}$.

3.4 Équivalence avec les expressions B-régulières

Une *expression B-régulière* (ou *B-expression*) respecte la syntaxe suivante :

$$E ::= \quad \varepsilon \quad | \quad a \quad | \quad E + E \quad | \quad E \cdot E \quad | \quad E^* \quad | \quad E^B \quad (|\emptyset) \ .$$

La sémantique $[\![e]\!]_B$ d'une expression régulière e est une application de A^* dans $\omega + 1$ définie par induction :

- $[\![\varepsilon]\!]_B(u) = \begin{cases} 0 & \text{si } u = \varepsilon \\ \omega & \text{sinon} \end{cases}$,
- $[\![a]\!]_B(u) = \begin{cases} 0 & \text{si } u = a \\ \omega & \text{sinon} \end{cases}$,
- $[\![f + g]\!]_B(u) = \min([\![f]\!]_B(u), [\![g]\!]_B(u))$,
- $[\![f \cdot g]\!]_B(u) = \min\limits_{u=vw} \max([\![f]\!]_B(u), [\![g]\!]_B(v))$,
- $[\![f^*]\!]_B(u) = \min\limits_{u=u_1\dots u_n} \max\limits_{i=1\dots n} [\![f]\!]_B(u_i)$,
- $[\![f^B]\!]_B(u) = \min\limits_{u=u_1\dots u_n} \max(n, \max\limits_{i=1\dots n} [\![f]\!]_B(u_i))$.

Exercice 10 *Montrer que pour toute B-expression E et tout K, $\{u \ : \ [\![E]\!]_B(u) \leq n\}$ est le langage obtenu en évaluant l'expression rationnelle obtenue de E en remplaçant toutes les occurrences de ^{B} par $^{\leq K}$.*

Proposition 10. *Si e est une expression régulière décrivant le langage L, alors*

$$[\![e]\!]_B(u) = \chi_L = \begin{cases} 0 & \text{si } u \in L \\ \omega & \text{sinon.} \end{cases}$$

Démonstration. Par récurrence sur l'expression.

Example 3. a^B, $b^*(ab^*)^B$, $b^*(ab^*)^B + a^*(ba^*)^B$, $(a^*b)^*a^B(ba^*)^*$.

Théorème 12 *Les expressions B-régulières sont $\approx$-équivalentes avec les B-automates.*

Démonstration. (principe) Des B-automates vers les B-expressions. Sans perte de généralité, on part d'un B-automate hiérarchique $\mathcal{A}$. La preuve est par récurrence sur le nombre de transitions de l'automate. Si l'automate n'a aucune transition, alors deux cas sont possibles : si il existe un état la fois initial et final, la fonction acceptée est $[\![\varepsilon]\!]_B$ sinon, c'est $[\![\emptyset]\!]_B$. Si l'automate contient une transition, alors on considère la transition (p, a, h, q) dont l'action h est maximale pour l'ordre $R_0 < I_1 < R_1 < \dots$. Soient $\mathcal{A}'_{I,F}$, $\mathcal{A}'_{I,p}$, $\mathcal{A}'_{q,p}$ et $\mathcal{A}'_{q,F}$ les automates obtenus de $\mathcal{A}$ en enlevant la transition (p, a, h, q), et en choisissant I ou $\{q\}$ comme états initiaux, et F ou $\{p\}$ comme états finaux en concordance avec l'indice de l'automate. Par hypothèse de récurrence, on obtient les B-expressions correspondantes $E'_{I,F}$, $E'_{I,p}$, $E'_{q,p}$ et $E'_{q,F}$. On pose :

$$E = \begin{cases} E'_{I,F} + E'_{I,p}a(E'_{q,p}a)^B E'_{q,F} & \text{si } h \text{ est de la forme } I_l \\ E'_{I,F} + E'_{I,p}a(E'_{q,p}a)^* E'_{q,F} & \text{sinon} \end{cases}$$

On montre sans difficulté que $[\![E]\!]_B \approx [\![\mathcal{A}]\!]_B$.

Des B-expressions vers les B-automates. La preuve est par récurrence sur la B-expression. La construction est particulièrement simple si on s'autorise les ϵ-transitions.

Le cas les plus intéressant sont quand l'expression est de la forme E^B. On applique l'hypothèse de récurrence sur un automate $\mathcal{A}$ qui calcule une fonction équivalence à $[\![E]\!]_B$. On construit un nouvel automate $\mathcal{A}'$ à partir de $\mathcal{A}$ en :

- ajoutant un état initial et final isolé,
- ajoutant un nouveau compteur γ (on peut améliorer la construction en choisissant un compteur γ déjà existant, pourvu que celui-ci n'ai jamais été remis à 0 dans $\mathcal{A}$),
- ajoutant pour tout état initial q et tout état final p une transition $(p, \epsilon, I_\gamma, q)$, où I_γ met à 0 tous les compteurs, sauf γ auquel l'action `ic` est attribuée.

Dans le cas d'une expression E^* la même construction est utilisée, à la différence près qu'aucun compteur n'est ajouté, et que les transitions rajoutées remettent tous les compteurs à zéro. On se convainc sans difficulté que cette construction transforme bien une B-expression en B-automate (en fait, hiérarchique) calculant la même fonction de coût.

3.5 Décidabilité de la divergence

Une fonction f de E dans $\omega + 1$ est *divergente* si pour tout n, l'ensemble $f^{-1}(n)$ est fini. En particulier, si $E = \omega$, alors f est divergente si et seulement si $\lim f = \omega$.

Fait 13 *Si $f \approx g$, alors f est divergente ssi g est divergente.*

Théorème 14 *Le problème de la divergence des fonctions calculées par B-automates est décidable.*

Démonstration. En utilisant le fait 13, on choisit sans perte de généralité une expression B-régulière. On étudie les propriétés suivantes :

- ε si son support est $\{\epsilon\}$, i.e., si elle est équivalente à $[\![\varepsilon]\!]_B$,
- NO si son support contient un mot non vide,
- ND si f est non divergente.

On alors les propriétés suivantes :
- si $e = \varepsilon$, alors e est ε,
- si $e = a$, alors e est NO,
- si $e = f + g$, alors e est la propriétés max de f et g pour l'ordre ε<NO<ND,
- si $e = f \cdot g$ alors e ... faire une table.
- si $e = f^*$ alors faire une table
- si $e = f^B$ alors faire une table.

□

4 Monoïdes de stabilisation

4.1 Définition

Un *semigroupe* $\langle S, \cdot \rangle$ est un ensemble S muni d'une opération de produit associative $(a \cdot (b \cdot c) = (a \cdot b) \cdot c)$. S'il contient un élément neutre 1, i.e., tel que $1 \cdot x = x \cdot 1 = x$ pour tout x, il est appelé monoïde. Un *semigroupe ordonné* $\langle S, \cdot, \leq \rangle$ est un semigroupe S équipé d'un ordre $\leq$ compatible, c.a.d, tel que $a \leq b$ et $a' \leq b'$ implique $a \cdot a' \leq b \cdot b'$. Un *idempotent* d'un semigroupe est un élément e tel que $e \cdot e = e$. L'ensemble des idempotents d'un semigroupe $\mathbf{S}$ est noté $E(\mathbf{S})$.

Definition 1. *Un* semigroupe de stabilisation $\langle S, \cdot, \leq, \sharp \rangle$ *est un semigroupe ordonné* fini $\langle S, \cdot, \leq \rangle$ *muni d'un opérateur* $\sharp : E(\mathbf{S}) \to E(\mathbf{S})$ *(appelé* stabilisation*) tel que :*
- *pour tous* $e \leq f$ *dans* $E(\mathbf{S})$, $e^\sharp \leq f^\sharp$;
- *pour tous* $a, b \in S$ *tels que* $a \cdot b \in E(\mathbf{S})$ *et* $b \cdot a \in E(\mathbf{S})$, *on a* $(a \cdot b)^\sharp = a \cdot (b \cdot a)^\sharp \cdot b$;
- *pour tout* $e \in E(\mathbf{S})$, $e^\sharp \leq e$;
- *pour tout* $e \in E(\mathbf{S})$, $(e^\sharp)^\sharp = e^\sharp$.

Un monoïde de stabilisiation *est un semigroupe de stabilisation dont le semigroupe sous-jacent est un monoïde, et tel que* $1^\sharp = 1$.

Example 4 («compter» les occurrences de la lettre a). On cherche à construire un monoïde de stabilisation de manière informelle, en ayant pour objectif de décrire la fonction de coût de la fonction calculant le nombre d'occurrences de la lettre a, sur l'alphabet $\{a, b\}$. En termes informels, on cherche à construire un monoïde de stabilisation qui sépare les mots contenants «beaucoup» d'occurrences de la lettre a, de ceux qui n'en contienne que «peu».

Pour atteindre cet objectif, il faut séparer 3 «types» de mots :
- les mots sans occurrences de a ; appelons ce type 'b' (la lettre b est de ce type).
- les mots contenant au moins une occurrence de a, mais seulement un petit nombre de telles occurrences ; appelons 'a' le type correspondant (en effet, le mot a est de ce type),
- les mots contenant un 'grand nombre' d'occurrences de a's' ; appelons ce type '0'.

Notre objectif est de séparer les mots de type 0, des mots de type a ou b. On pose donc $M = \{a, b, 0\}$.

Bien entendu, répéter un peu ou beaucoup des mots qui ne contiennent aucun a ne peut produire que des mots qui ne contiennent aucun a. Ce phénomène s'obtient en posant $b \cdot b = b$ et $b^\sharp = b$.

Itérer un petit nombre de mots contenant un petit nombre de a (au moins 1) produit un mot de même nature, ainsi, on pose $a \cdot a = a$. De même, concaténer de tels mots avec des mots sans a, à gauche comme à droite, ne change pas le nombre de a. On pose donc $a \cdot b = b \cdot a = a$. En revanche, itérer un grand nombre de mots contenant au moins une occurrence de la lettre a produit un mot qui contient un grand nombre de a. On pose donc $a^\sharp = 0^\sharp = 0$.

En utilisant des arguments informels similaires, on produit également les règles $a \cdot 0 = 0 \cdot a = 0 \cdot 0 = 0$. Cela se résume au moyen du tableau suivant.

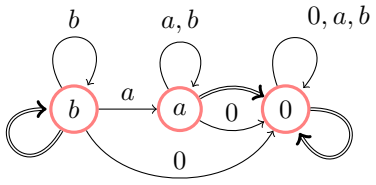
	b	a	0	$\sharp$
b	b	a	0	b
a	a	a	0	0
0	0	0	0	0

La partie gauche du tableau représente le produit, la colonne supplémentaire donne la table de stabilisation (remarquons que dans cette notation, il se peut que cette colonne ne soit que partiellement définie, car la stabilisation n'est définie que sur les idempotents.).

Pour compléter cette définition, il reste à décrire la relation d'ordre. Par définition d'un monoïde de stabilisation, il est nécessaire de poser $0 = a^\sharp \leq a$. On peut vérifier que cette relation (augmentée de $x \leq x$ pour tout x) aboutit bien à la définition d'un monoïde de stabilisation.

L'idée qu'il faut avoir de cet ordre est qu'il précise comment la valeur d'un mot peut évoluer quand le curseur séparant «un peu» de «beaucoup» varie. Par exemple, le mot a^{100} doit être considéré comme petit si ce curseur est fixé à 1000, et donc doit avoir la valeur a. En revanche, si l'on fixe le curseur sur 10, ce même mot doit être considéré comme ayant beaucoup d'occurrences de la lettre a, et donc doit avoir la valeur 0. Ainsi, il y a une sorte de «continuum» entre la valeur a et la valeur 0, puisqu'un même mot peut passer de l'un à l'autre quand le curseur varie.

Il est agréable également de représenter le monoïde de stabilisation au moyen de son graphe de Cayley :



Chaque sommet représente l'un des éléments du monoïde, et un arc étiqueté y relie chaque sommet x au sommet $x \cdot y$. Une double flèche relie chaque idempotent x à son «stabilisé» $x^\sharp$.

Remark 6. Un monoïde au sens usuel $\langle M, \cdot \rangle$ se transforme naturellement en un monoïde de stabilistion $\langle M, \cdot, \sharp, \leq \rangle$, dans lequel la stabilisation de tout idempotent e est $e^\sharp = e$, et l'ordre est réduit à l'identité. On vérifie sans difficulté que cette définition satisfait bien toutes les contraintes d'un monoïde de stabilisation.

Exercice 11 *Construire les monoïdes de stabilisations qui «mesurent»*
- *la taille du plus grand segment de a consécutifs,*
- *la taille du plus petit segment de a consécutifs,*
- *sur $\{a, b, c\}$, le minimum du nombre de a et du nombre de b,*
- *le nombre d'occurrences du facteur ab dans un mot.*

4.2 Vers une sémantique des monoïdes de stabilisation : les n-calculs

On se fixe un monoïde de stabilisation $\langle M, \cdot, \sharp, \leq \rangle$.

Soit $u \in M^*$. On cherche à calculer la valeur de u dans le monoïde de stabilisation, pour une valeur de n fixée, où n détermine quelle est la limite entre «peu» et «beaucoup». Pour cela, il s'agit de formaliser la notion de calcul : un calcul est un arbre décrivant les étapes successives aboutissant à un résultat.

Definition 2. *Un* n-(arbre de) calcul sur le mot $u \in M^+$ *est un arbre d'arité non fixée, ordonné, dont chaque nœud x est étiqueté par un élément $v(x) \in M$ appelé sa* valeur *et tel que :*
- *les étiquettes des feuilles lues de gauche à droite forment le mot u,*
- *pour chaque noeud x, de fils $y_1, \ldots, y_k$ (de gauche à droite), l'une de ces situations est vraie :*
  - *$k = 0$ (feuille),*
  - *$k = 2$, et $v(x) = v(y_1) \cdot v(y_2)$, (nœud produit),*
  - *$2 < k \leq n$, et $v(x) = v(y_1) = \cdots = v(y_k) \in E(M)$, (nœud idempotent),*
  - *$k > n$, $v(y_1) = \cdots = v(y_k) = e \in E(M)$ et $v(x) = e^\sharp$ (nœud de stabilisation).*

La valeur *d'un abre de calcul est la valeur de sa racine.*

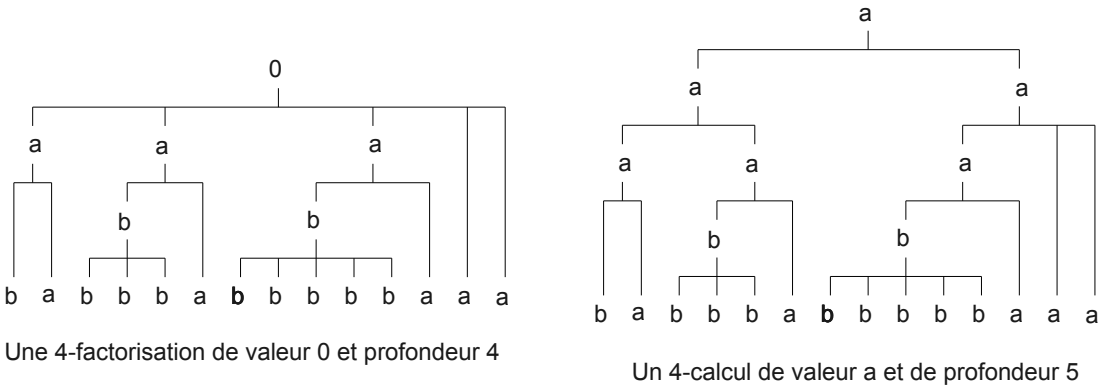


Figure 1. Deux arbres de 4-calcul sur le mot *babbbabbbbbaaa* dans le monoïde comptant le nombre d'occurrences de a.

Dans les preuves, on notera $[a]$ le calcul restreint à une feuille de valeur a. On notera $[a](t_1 \cdot t_2)$ le calcul de valeur a formé d'un nœud produit, et dont les sous-arbres des fils sont les calculs t_1 et t_2. On utilisera également la notation similaire $[a](t_1 \dots t_k)$ pour dénoter les nœuds idempotents et de stabilisations. Ces notations seront également utilisées pour les notions variantes de sous-calculs et sur-calculs.

Deux exemples de calculs sont présentés à la figure 1. L'idée essentielle est qu'un calcul représente un calcul aboutissant à sa valeur comme résultat. Ainsi, étant donné un mot $u \in M^+$, et n on aimerait définir la n-valeur d'un mot u dans M^+ comme la valeur d'un des n-calculs sur u.

On doit alors résoudre les deux questions suivantes :

– Est-ce que tout mot admet un n-calcul pour tout n ?

– Est-ce que deux n-calculs d'un même mot ont la même valeur ?

La réponse à la première question est à priori élémentaire : il suffit de n'utiliser que des nœuds binaires, et de construire une factorisation sur ce principe. Cette solution n'est bien évidemment pas satisfaisante. En effet, elle ne capture pas la sémantique que l'on cherche puisque aucun nœud utilisant une stabilisation n'est créé de la sorte. Ce problème vient du fait que l'on cherche, en même temps, à contrôler un autre paramètre : la profondeur du calcul de calcul (c.a.d, sa hauteur). On peut remarquer qu'à partir du moment ou la profondeur est bornée, alors que la longueur des mots ne l'est pas, on se retrouve obligé pour construire un calcul, d'utiliser des nœuds impliquant au moins une stabilisation.

La réponse à la première question reste positive, mais cette fois-ci requiert l'utilisation d'arguments plus subtils ; en l'espèce, il faut recourir aux relations de Green. Il s'agit en fait d'une variation autour d'un résultat de Simon sur les monoïdes, le théorème des forêts de factorisations [19], dont il existe plusieurs autres preuves [16,4,3].

Lemme 15 (admis) *Pour tout $u \in M^+$ et tout n, il existe un n-calcul de profondeur au plus $3|M|$.*

La réponse à la seconde question est négative, comme le montre les deux exemples de 4-calculs de la figure 1. il est possible qu'un même mot admette deux n-calculs de valeurs différentes. Cela provient du fait que les énoncés informels du type : «peu + peu = peu» ne peuvent être vrai dans l'absolu pour une valeur limite fixée n. En revanche, il est possible de montrer que les résultats de différents calculs ne se contredisent pas, modulo une certaine approximation. On va en fait démonter un résultat un peu plus fort, mettant en jeu des variations sur la notion de calcul.

Definition 3. *Un* n-sous-calcul pour le mot u *est un arbre d'arité non fixée, ordonné, dont chaque nœud x est étiqueté par un élément $v(x) \in M$ appelé sa* valeur *et tel que :*

- *les étiquettes des feuilles lues de gauche à droite forment un mot $\leq u$ (l'ordre sur M est étendu aux mots de M^* lettre à lettre),*
- *pour chaque noeud x, de fils $y_1, \ldots, y_k$ (de gauche à droite), l'une de ces situations est vraie :*
  - *$k = 0$ (i.e, x est une feuille),*
  - *$k = 2$, et $v(x) \leq v(y_1) \cdot v(y_2)$,*
  - *$2 < k \leq n$, et $v(x) \leq v(y_1) = \cdots = v(y_k) = v(x) \in E(M)$,*
  - *$k > n$, $v(y_1) = \cdots = v(y_k) = e \in E(M)$ et $v(x) \leq e^\sharp$.*

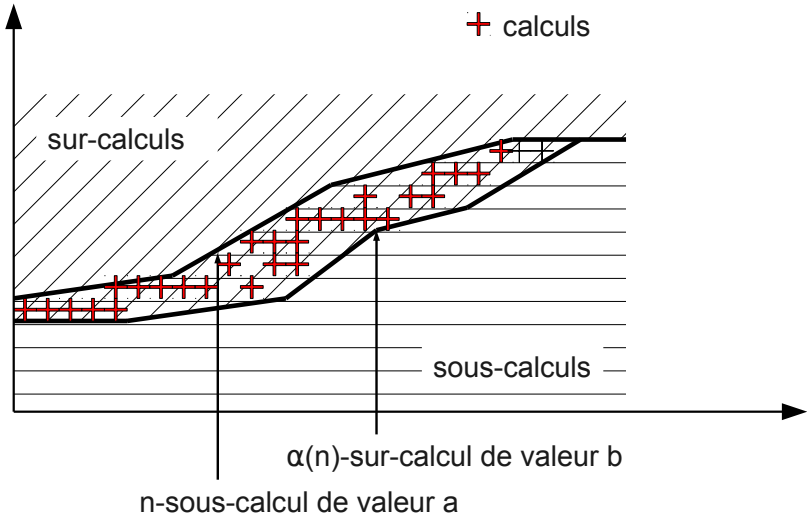
Un n-sur-calcul *est défini identiquement en remplaçant $\leq$ par $\geq$ partout.*

La valeur *d'un arbre de sur/sous-calcul est la valeur de sa racine.*

Idée : les sous-calculs permettent de sous-approximer la valeur d'un mot, les sur-calculs permettent de le sur-approximer. Les sous-calculs et les sur-calculs ont des propriétés que les calculs n'ont pas et qui les rendent plus faciles à manipuler, comme le montre le fait suivant :

Fait 16 *Tout n-calcul est un n-sous-calcul et un n-sur-calcul.*
Soit $m \leq n$, alors tout m-sous-calcul est est n-sous-calcul, et tout n-sur-calcul est un m-sur-calcul.

Cela se s'interprète sur le dessin suivant.



Un mot est fixé. Les abscisses correspondent à la valeur de n, et les ordonnées à l'ordre (idéalisé) dans le monoïde de stabilisation. Les n-calculs de valeur a (que l'on suppose de profondeur constante égale, par exemple, à $3|M|$), sont symbolisés par des croix rouges à l'abscisse n et l'ordonnée a. Les sous-calculs, qui sont clos vers le bas et la droite d'après le lemme précédent, délimitent la zone rayée horizontalement. Les sur-calculs, sont eux clos vers la gauche et le haut, et correspondent à la zone rayée obliquement. Le lemme suivant énonce que les frontières de ces zones, en gras sur le dessin, ne peuvent être arbitrairement éloignées ; elles se comportent, globalement, de manière identique.

Lemme 17 (admis) *Pour tout p, il existe α telle que pour tout $\alpha(n)$-sur-calcul de valeur b sur un mot u de profondeur au plus p et tout n-sous-calcul sur le même mot de valeur a de profondeur au plus p, $a \leq b$.*

Ainsi, il faut que la profondeur soit suffisamment grande pour qu'il existe un calcul par le Lemme 15, mais bornée pour qu'un unique α permette de borner la marge d'erreur par le Lemme 17.

Example 5. Construire des arbres de calculs sur les monoïdes de stabilisation exemple.

Example 6. Un monoïde $\langle M, \cdot \rangle$ peut être transformé en un monoïde de stabilisation $\langle M, \cdot, id, = \rangle$. Alors, pour tout n, un n-calcul (resp. sur/sous-calcul) sur un mot u a toujours pour valeur $\pi(u)$.

Reconnaissabilité

Soit M un monoïde de stabilisation. Un *idéal* de M est une partie $I \subseteq M$ telle que $a \leq b \in I$ implique $a \in I$.

Lemme 18 *Soit un monoïde de stabilisation* $\mathbf{M}$ *une application* $h : \mathbb{A} \to M$*, et un idéal* $I \subseteq M$*, on pose :*

$$[\![\mathbf{M}, h, I]\!]_p^+(u) = \sup\{n+1 \ : \ \textit{il existe un } n\textit{-sur-calcul de valeur dans } I \textit{ de prof au plus } p \textit{ sur } h(u)\}$$

$$[\![\mathbf{M}, h, I]\!]_p^-(u) = \inf\{n \ : \ \textit{il existe un } n\textit{-sous-calcul de valeur dans } M \setminus I \textit{ de prof au plus } p \textit{ sur } h(u)\}$$

Alors :
- *Les fonctions* $[\![\mathbf{M}, h, I]\!]_p^+$ *et* $[\![\mathbf{M}, h, I]\!]_p^-(u)$ *sont croissantes (en fonction du mot en entrée, ordonné lettre-à-lettre),*
- *Si* $p \geq 3|M|$*, alors* $[\![\mathbf{M}, h, I]\!]_p^- \leq [\![\mathbf{M}, h, I]\!]_p^+$*,*
- *Pour tout* p*, il existe* α *tel que* $[\![\mathbf{M}, h, I]\!]_p^+ \leq \alpha([\![\mathbf{M}, h, I]\!]_p^-)$*.*
- *Pour tout* $p \leq r$*,* $[\![\mathbf{M}, h, I]\!]_r^- \leq [\![\mathbf{M}, h, I]\!]_p^-$ *et* $[\![\mathbf{M}, h, I]\!]_p^+ \leq [\![\mathbf{M}, h, I]\!]_r^+$*.*

Démonstration. Le premier item utilise le fait 16 et le fait que I est un idéal. Le second item est une conséquence du lemme 15 en combinaison avec le fait 16 : l'existence d'un n-calcul témoigne de l'inégalité. Le troisième item est une conséquence du lemme 17 : sous-calculs et sur-calculs ne se contredisent pas. Le dernier item correspond au fait que plus p est grand, plus il y a de sur/sous-calculs de profondeur au plus p.

Ainsi, toutes les fonctions $[\![\mathbf{M}, h, I]\!]_p^+$ et $[\![\mathbf{M}, h, I]\!]_p^-$ sont équivalentes pour $\approx$. Leur classe d'équivalence est appelée *fonction de coût reconnue par* $\mathbf{M}, h, I$.

4.3 ♯-expression

Une *♯-expression* sur A est une expression composée de lettres dans A, de produits, et d'exposant par ♯. Une ♯-expression sur un monoïde de stabilisation $\mathbf{M}$ est une ♯-expression sur M. Elle est *bien typée* si elle dénote un calcul valide dans le monoïde de stabilisation, c'est à dire qu'il est possible de l'évaluer dans $\mathbf{M}$ en utilisant les opérations de produit et de stabilisation de $\mathbf{M}$. Le résultat de ce calcul est appelé *valeur* de la ♯-expression. Une ♯-expression est *stricte* si elle contient au moins une occurrence de ♯.

Le *n-dépliage* se définit par récurrence comme :

$$\begin{aligned} \text{depliage}(a, n) &= a \\ \text{depliage}(EF, n) &= \text{depliage}(E, n)\text{depliage}(F, n) \\ \text{depliage}(E^\sharp) &= \overbrace{\text{depliage}(E, n) \dots \text{depliage}(E, n)}^{n+1 \text{ fois}} \end{aligned}$$

Dans le cas où E est bien typée, pour $n \geq 3$, le *n-calcul de E* est définit par récurrence comme l'arbre de calcul

$$\begin{aligned} \text{calcul}(a, n) &= a \\ \text{calcul}(EF, n) &= [v(EF)](\text{calcul}(E, n), \text{calcul}(F, n)) \\ \text{calcul}(E^\sharp) &= [v(E^\sharp)](\overbrace{\text{calcul}(E, n), \dots, \text{calcul}(E, n)}^{n+1 \text{ fils}}) \end{aligned}$$

Bien entendu, calcul(E,n) est un n-calcul sur le mot depliage(u,n), de valeur $v(E)$. En fait, c'est un peu plus fort :

Fait 19 calcul(E,n) *est un* m*-calcul sur le mot* depliage(E,n) *pour tout* $m \leq n$.

4.4 Produit de monoïdes de stabilisation

Soient $M = \langle M, \cdot, \sharp, \leq \rangle$ and $M' = \langle M', \cdot', \sharp', \leq' \rangle$ deux monoïdes de stabilisation. On définit leur produit comme :

$$M \times M' = \langle M \times M', \cdot'', \sharp'', \leq'' \rangle$$

où $(a,a') \cdot'' (b,b') = (a \cdot b, a' \cdot' b')$, $(e,e')^{\sharp''} = (e^{\sharp}, e'^{\sharp'})$, et $(a,a') \leq'' (b,b')$ ssi $a \leq b$ et $a' \leq' b'$.

Étant donnés deux mots u, u' sur M et M' respectivement, de même longueur, on définit $u \times u'$ comme le mot de $M \times M'$ qui projeté sur M et M' donne u et u' respectivement.

Lemme 20 *S'il existe un* n*-calcul (resp. sous-calcul, resp. sur-calcul) sur* $u \times v$ *de valeur* (a,b) *dans* $M \times M'$*, alors il existe un* n*-calcul (resp. sous-calcul, resp. sur-calcul) sur* u *dans* M *de valeur* a *et un* n*-calcul (resp. sous-calcul, resp. sur-calcul) sur* v *dans* M' *de valeur* b.

Démonstration. Par projection des valeurs.

Corollaire 21 *Soient deux fonctions de coût reconnaissables* f, g *sur un même alphabet* $\mathbb{A}$*, alors il existe un monoïde de stabilisation* $\mathbf{M}$*, une application de* $\mathbb{A}$ *dans* M *et deux idéaux* I, J *tels que* $\mathbf{M}, h, I$ *reconnait* f *et* $\mathbf{M}, h, J$ *reconnait* g.

Corollaire 22 *Les fonctions reconnaissables sont closes sous* min *et* max.

4.5 Décidabilité de la l'existence d'une borne

Théorème 23 *La relation* $\preccurlyeq$ *est décidable sur les fonctions de coût reconnaissables.*

Démonstration. Soient deux fonctions de coût régulières f, g. D'après le corollaire 21 il existe un monoïde M, une application h de l'alphabet $\mathbb{A}$ dans M, et deux idéaux I, J tels que $\mathbf{M}, h, I$ reconnaisse f et $\mathbf{M}, h, J$ reconnaisse g.

On établit que $f \leq g$ ssi pour tout $a \in \langle h(\mathbb{A}) \rangle^*$, $a \in I$ implique $a \in J$.

$\Leftarrow$ Supposons pour tout $a \in \langle h(\mathbb{A}) \rangle^*$ que $a \in I$ implique $a \in J$. Soit u un mot sur $\mathbb{A}$, et t un n-arbre de calcul sur u, de valeur a. Bien entendu, $a \in \langle h(\mathbb{A}) \rangle^*$, et donc si $a \in I$ alors $a \in J$. On en déduit que :

$$f \approx \sup\{n \ : \ n\text{-calcul sur } u \text{ de valeur } a \in I\} \leq \sup\{n \ : \ n\text{-calcul sur } u \text{ de valeur } a \in J\} \approx g.$$

$\Rightarrow$ Supposons qu'il existe $a \in \langle h(\mathbb{A}) \rangle^*$ tel que $a \in I$ mais $a \notin J$. Il existe une $\sharp$-expression bien typée E de valeur a sur l'alphabet $h(\mathbb{A})$. Soit E' obtenue de E en substituant à tout $x \in \mathbb{A}$ une lettre b telle que $h(b) = x$. Or, calcul(E,n) est un n-calcul sur $u_n = \text{depliage}(E,n) = h(\text{depliage}(E',n))$ de profondeur P et de valeur $a \in I$. Donc $[\![\mathbf{M}, h, I]\!]_P^+(u_n) \geq n$. Mais calcul$(E,n)$ est aussi un 0-calcul sur u_n, de valeur $a \notin J$. Donc $[\![\mathbf{M}, h, J]\!]_P^+(u_n) = 0$.

4.6 Clôture sous inf-projection : le monoïde des idéaux

Soit $\mathcal{I}$ l'ensemble des idéaux sur M. On équipe $\mathcal{I}$ d'une structure d'ordre par :

$$A \leq B \qquad \text{iff} \quad A \subseteq B \ ,$$

d'un produit par :

$$A \cdot B \quad = \quad \{c \leq a \cdot b \ : \ a \in A, \ b \in B\}$$

et d'un opérateur de stabilisation par :

$$E^{\sharp} = \{x \leq a \cdot e^{\sharp} \cdot b \ : \ a, b, e \in E, \ e \cdot e = e\} \ .$$

Lemme 24 *Si E est idempotent, alors tout $a \in E$ est tel que $a \leq b \cdot e \cdot c$ où $b, c, e \in E$, e idempotent.*

Démonstration. Comme $E = E \cdots E$, pour tout n, il existe $a_1, \ldots, a_n \in E$ tels que $a \leq a_1 \cdots a_n$. Par Ramsey, pour n suffisament grand, il existe i, j tels que $a_i \cdots a_{j-1} = e$ est idempotent et $i > 1$. On pose alors $b = a_1 \cdots a_{i-1}$, $c = a_j \cdots a_n$. On a bien $b, c, e \in E$, et $a \leq b \cdot e \cdot c$.

Lemme 25 *$M_{\mathcal{I}} = \langle \mathcal{I}, \cdot, \sharp, \subseteq \rangle$ est un monoïde de stabilisation.*

Lemme 26 *Pour tout idéal idempotent E, et tout sous-calcul t sur un mot $u \in E^+$ de valeur a, $a \in E$. Si t contient un nœud de stabilisation, alors $a \in E^\sharp$.*

Démonstration. Par récurrence sur la profondeur du calcul.

Lemme 27 *Tout n-calcul de profondeur p sur un mot de longueur au moins $n^p + 1$ contient un nœud de stabilisation.*

Démonstration. Si le calcul ne contient aucun nœud de stabilisation, alors tous ses nœuds ont une arité au plus n. Par conséquent, un tel calcul de profondeur p ne peut avoir plus de n^p feuilles. Contradiction.

Lemme 28 *Soient $A_1 \ldots A_k$ un mot sur $\mathcal{I}$ et un n-calcul T sur $A_1 \ldots A_k$ de profondeur p et de valeur A. Pour tout $a \in A$, il existe un n-sous-calcul de profondeur $3p$ de valeur a sur un mot $a_1 \ldots a_k$ tel que $a_1 \in A_1, \ldots, a_k \in A_k$.*

Démonstration. On étudie le cas des profondeurs 0 et 1 en premier lieu.

feuille $T = [A_1]$ Soit $a \in A = A_1$, $[a]$ est un n-calcul de valeur $a \in A$.

produit $T = [A]([A_1] \cdot [A_2])$. Soit $a \in A$, par définition du produit, il existe $a_1 \in A_1$ et $a_2 \in A_2$ tels que $a \leq a_1 \cdot a_2$. Le n-sous-calcul $[a]([a_1] \cdot [a_2])$ convient.

idempotent $T = [E](\overbrace{[E] \ldots [E]}^{k})$ avec $k \leq n$. Soit $a \in E$. On a $a \leq b \cdot e \cdot c$ pour $b, c, e \in E$, e idempotent (lemme 24). On construit le n-sous-calcul $[a]([b \cdot e]([b] \cdot [e](\overbrace{[e] \ldots [e]}^{k-2})) \cdot [c])$.

stabilisation $T = [E^\sharp](\overbrace{[E] \ldots [E]}^{k})$ (avec $k > n$). Soit $a \in E^\sharp$. a s'écrit $a \leq b \cdot e^\sharp \cdot c$ pour $b, e, c \in E$, e idempotent. On construit le n-sous-calcul$[a]([b \cdot e^\sharp]([b] \cdot [e^\sharp](\overbrace{[e] \ldots [e]}^{k-2})) \cdot [c])$.

Le cas général s'obtient par une application récursive des cas précédents.

Lemme 29 *Soient $A_1 \ldots A_k$ un mot sur $\mathcal{I}$ et un $n^{3|M|}$-calcul T sur $A_1 \ldots A_k$ de profondeur p de valeur A. Pour tout mot $u = a_1 \ldots a_k \in A_1 \ldots A_k$ (c.à.d. $a_1 \in A_1$, …, $a_k \in A_k$), il existe un n-calcul sur $a_1 \ldots a_k$ de valeur $a \in A$ et de profondeur au plus $3|M|p$.*

Démonstration. Considérons tout d'abord le cas des calculs de profondeur 0 et 1. Il y a 4 cas :

feuille $T = [A]$, et $u = a \in A$, donc $[a]$ est un calcul qui convient.

produit $T = [A]([B] \cdot [C])$ auquel cas, $u = bc$ pour $b \in B$ et $c \in C$. On construit le n-calcul $[b \cdot c]([b] \cdot [c])$ de valeur $b \cdot c \in A$.

idempotent $T = [E](\overbrace{[E] \ldots [E]}^{k})$ (pour $k \leq n^{3|M|}$). Soit t un n-calcul sur $u = a_1 \ldots a_k$ de profondeur au plus $3|M|$ de valeur a (il existe d'après le lemme 15). Cet arbre satisfait $a \in E$ d'après le lemme 26.

stabilisation $T = [E^\sharp](\overbrace{[E] \ldots [E]}^{k})$ pour $k > n^{3|M|}$. Soit t un n-calcul sur $u = a_1 \ldots a_k$ de profondeur au plus $3|M|$ de valeur a (il existe d'après le lemme 15). D'après 27, il possède un nœud de stabilisation. Donc sa valeur est dans $E^\sharp$ d'après le lemme 26.

La preuve pour tout p est par récurrence sur la profondeur du calcul T, en appliquant les 4 cas ci-dessus.

Théorème 30 *Les fonctions de coût reconnaissables sont closes sous inf-projection.*

Démonstration. On suppose la fonction f sur l'alphabet $\mathbb{A}$ reconnue par $\mathbf{M}, h, I$. Soit z application de $\mathbb{A}$ vers $\mathbb{B}$. On construit le monoïde de stabilisation des idéaux $M_{\mathcal{I}}$, l'application H envoyant $b \in \mathbb{B}$ sur $\{x \ : \ x \leq h(a) \ : \ z(a) = b\}$, et en considérant l'idéal K de $M_{\mathcal{I}}$:

$$K = \{J \in \mathcal{I} \ : \ J \subseteq I\} \ .$$

Il s'agit de montrer que cette construction est correcte. Soit F la fonction reconnue par $M_{\mathcal{I}}, H, K$.

Soit u un mot sur $\mathbb{B}$, et $n = F(u)$, i.e.,

$$F(u) = \inf\{n \ : \text{ il existe un } n\text{-sous-calcul sur } H(u) \text{ de valeur } A \not\subseteq I\} \ .$$

Considérons la valeur $A \not\subseteq I$ témoignant de la valeur n. Soit $a \in A \setminus I$ et T le n-sous-calcul correspondant. Ce n-sous-calcul peut être transformé en un n-sous-calcul de M de valeur a sur un mot $v \in H(u)$ par le lemme 28. On en déduit que $n \geq f(v) \geq \inf\{f(v) \ : \ z(v) = u\}$.

Réciproquement, fixons u, et soit $n = \inf\{f(v) \ : \ z(v) = u\} + 1$ (sup-def de f pour $3|M|^2$). Par définition, il existe v tel que $z(v) = u$, tel que tout n-sous-calcul sur $h(v)$ a une valeur $a \notin I$ (de prof $3|M|^2$). Considérons un n^P-calcul sur $H(u)$ de valeur A de hauteur P. Par le Lemme 29, il existe un n-calcul de valeur $a \in A$, et de profondeur au plus P^2. On en déduit que $a \notin I$. Donc $A \in K$. Donc $F(u) \leq n$.

4.7 Des monoïdes de stabilisation vers les automates

On construit une B-expression qui donne à un mot une valeur plus petite que n s'il existe un n-sous-calcul sur ce mot. La construction est par induction sur la profondeur p du calcul.

$$E_{a,0} = \emptyset \qquad E_{a,p+1} = \sum_{b \in \mathbb{A},\ h(b) \geq a} b + \sum_{b \cdot c \geq a} E_{b,p} E_{c,p} + \sum_{e = e \cdot e \geq a} E^B_{a,p+1} + \sum_{e^\sharp \geq a} E^*_{b,p+1} \ .$$

On pose enfin :

$$E = \sum_{a \notin I} E_{a,3|M|} \ .$$

On montre par récurrence que $E_{a,p} = [\![\mathbf{M}, h, \{x \leq a\}]\!]^+_p$, d'où l'on déduit $[\![E]\!]_B = [\![\mathbf{M}, h, I]\!]^+_{3|M|}$.

4.8 Des automates vers les monoïdes de stabilisation

A FAIRE.

5 *S*-automates et théorème de dualité

Les S-automates ressemblent de beaucoup de points de vue aux B-automates, bien qu'ils leur soient en un sens opposés. Les résultats vus précédemment pour les B-automates ont tous une contrepartie concernant les S-automates. Nous ne démontrons pas ces résultats dans ce support de cours. Les preuves suivent des schémas similaires, mais doivent toutes être entièrement reconstruites.

Le résultat le plus important est le théorème 33 qui énonce que les fonctions calculées par B-automates et par S-automates sont équivalentes.

5.1 Définition des S-automates

5.2 Coût d'une séquence

Considérons une séquence $u \in \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{c}\}^*$, on a vu lors de la définition des B-automates l'ensemble $C(u)$ qui collecte les valeurs observées durant la séquence. Cette fois-ci, on pose :

$$\mathit{coût}_S(u) = \min C(u) \ .$$

Le coût d'une séquence est la plus petite valeur prise par le compteur au moment d'être testé, juste avant d'être remis à 0.

Soit Γ un ensemble fini de *compteurs*, l'ensemble des *actions sur Γ* est $A_\Gamma = \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{cr}\}^\Gamma$. Soit :

$$\begin{array}{rl} \mathit{coût}_S : & A_\Gamma^* \to \omega \\ & u \to \min C(u) \end{array}$$

(en particulier $\mathit{coût}_S = \omega$ si $\Gamma = \emptyset$).

Un *S-automate* $\mathcal{A} = \langle Q, A, I, F, \Gamma, \Delta \rangle$ consiste en :
- un ensemble fini d'*états* Q,
- un *alphabet* fini A,
- un ensemble d'états *initiaux* $I \subseteq Q$,
- un ensemble d'états *finaux* $F \subseteq Q$,
- un ensemble de transitions $\Delta \subseteq Q \times A \times A_\Gamma \times Q$.

Une exécution ρ de $\mathcal{A}$ sur $u \in A^*$ est une séquence $\rho = (p_0, a_1, t_1, p_1) \dots (p_{n-1}, a_n, t_n, p_n)$ telle que :
- $(p_{i-1}, a_i, t_i, p_i) \in \Delta$,
- $a_1 \dots a_n = u$,
- $p_0 \in I$,
- $p_n \in F$.

Son coût est $\mathit{coût}_S(\rho) = \mathit{coût}_S(t_1 \dots t_n)$.

On définit le *coût* d'un mot $u \in A^*$ pour $\mathcal{A}$ comme :

$$[\![\mathcal{A}]\!]_S(u) = \max\{\mathit{coût}_S(\rho) \ : \ \rho \text{ exécution de } \mathcal{A} \text{ sur } u\} \qquad \text{(calculé dans } \omega + 1).$$

Remark 7. Si il n'y a aucune exécution de $\mathcal{A}$ sur un mot u, alors $[\![\mathcal{A}]\!]_S(u) = 0$.

Remark 8. Si $\mathcal{A}$ est un S-automate sans compteur (i.e., $\Gamma = \emptyset$) alors $\mathcal{A}$ peut être vu comme un automate non-déterministe acceptant le langage L. On a alors pour tout mot u :

$$[\![\mathcal{A}]\!]_S(u) = \begin{cases} \omega & \text{si } u \in L, \\ 0 & \text{sinon.} \end{cases}$$

En particulier $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ ssi $\mathcal{A} \leq \mathcal{A}$.

Cette situation dénote un comportement des S-automates opposé à celui des B-automates.

Example 7. Nombre de a.

Example 8. Max du nombre de a et de b.

Example 9. Min du nombre de a et de b.

Example 10. Taille du plus petit segment maximal formé uniquement de a.

5.3 Résultats élémentaires

Les résultats élémentaires sont similaires au cas des B-automates, les constructions sont identiques.

Proposition 11. *Le support d'une fonction définie par B-automate est régulier (i.e., l'ensemble des mots de valeur finie).*

Proposition 12. *Les fonctions calculées par B-automate sont closes sous* min, max *et* sup*-projection (soit f un morphisme lettre à lettre, et $\mathcal{A}$ un B-automate, il existe $\mathcal{A}'$ tel que $[\![\mathcal{A}']\!]_S(u) = \sup\{[\![\mathcal{A}]\!]_S(v) \ : \ f(v) = u\}$).*

5.4 Équivalences

Tout comme pour les B-automates, il existe plusieurs formalismes équivalents aux S-automates modulo $\approx$.

Théorème 31 *Les propriétés suivantes sont effectivement équivalentes :*

1. *f est calculée par un S-automate (avec ou sans ϵ-transitions) modulo $\approx$,*
2. *f est calculée par un S-automate hiérarchique (avec ou sans ϵ-transitions) modulo $\approx$,*
3. *f est calculée par une expression S-régulière modulo $\approx$ ou est bornée.*

Les définitions des S-automates hiérarchiques, et des S-automates avec ϵ-transitions sont similaires à celle des B-automates (il faut remplacer inf en sup dans le dernier cas). Il reste à définir les expressions S-régulières.

Une *expression S-régulière* (ou *S-expression*) respecte la syntaxe suivante :

$$E ::= \quad \varepsilon \quad | \quad a \quad | \quad E + E \quad | \quad E \cdot E \quad | \quad E^* \quad | \quad E^S \quad (|\emptyset) \ .$$

La sémantique $[\![e]\!]_S$ d'une expression S-régulière e est une application de $\mathbb{A}^*$ dans $\omega+1$ définie par induction :

- $[\![\varepsilon]\!]_S(u) = \begin{cases} \omega & \text{si } u = \varepsilon \\ 0 & \text{sinon} \end{cases}$,
- $[\![a]\!]_S(u) = \begin{cases} \omega & \text{si } u = a \\ 0 & \text{sinon} \end{cases}$,
- $[\![f + g]\!]_S(u) = \max([\![f]\!]_S(u), [\![g]\!]_S(u))$,
- $[\![f \cdot g]\!]_S(u) = \max\limits_{u=vw} \min([\![f]\!]_S(u), [\![g]\!]_S(v))$,
- $[\![f^*]\!]_S(u) = \max\limits_{u=u_1 \dots u_n} \min\limits_{i=1\dots n} [\![f]\!]_S(u_i)$,
- $[\![f^S]\!]_S(u) = \max\limits_{u=u_1 \dots u_n} \min(n, \min\limits_{i=1\dots n} [\![f]\!]_S(u_i))$.

Proposition 13. *Si e est une expression régulière décrivant le langage L, alors*

$$[\![e]\!]_S(u) = \begin{cases} \omega & \textit{si } u \in L \\ 0 & \textit{sinon.} \end{cases}$$

Example 11. $[\![a^S]\!]_S = [\![a^B + (a^*b)^+a^*]\!]_B$, $[\![(a^Sb)^*]\!]_S = [\![(a^*b)^*a^Bb(a^*b)]\!]_B$,

Théorème 32 *L'existence d'une borne est décidable pour les fonctions calculées par S-automates.*

Démonstration. On part d'une expression S-régulière. On simplifie les $\emptyset$. Si le résultat est $\emptyset$, alors l'expression est bornée.

5.5 Théorème de dualité, et fonctions régulières de coût

Le cœur de la théorie repose sur le théorème suivant.

Théorème 33 (dualité [5]) *Les propriétés suivantes sont équivalentes pour une fonction f de $\mathbb{A}^*$ dans $\omega+1$:*

- *f est calculée par un B-automate modulo $\approx$,*
- *f est calculée par un S-automate modulo $\approx$.*

La preuve de ce théorème nécessite de passer par le modèle équivalent des fonctions de coût reconnaissables par monoïde de stabilisation. Elle n'est pas donnée dans ces notes.

Références

1. Sebastian Bala. Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. In *STACS*, volume 2996 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2004.
2. Achim Blumensath, Martin Otto, and Mark Weyer. Boundedness of monadic second-order formulae over finite words. In *36th ICALP*, Lecture Notes in Computer Science, pages 67–78. Springer, July 2009.
3. Mikolaj Bojanczyk. Factorization forests. In *Developments in Language Theory*, volume 5583, pages 1–17, 2009.
4. Thomas Colcombet. Factorisation forests for infinite words. In *FCT 07*, number 4639 in Lecture Notes in Computer Science, pages 226–237. Springer, 2007.
5. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *36th ICALP*, number 5556 in Lecture Notes in Computer Science, pages 139–150. Springer, 2009.
6. Françoise Dejean and Marcel-Paul Schützenberger. On a question of Eggan. *Information and Control*, 9(1) :23–25, 1966.
7. Lawrence C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10 :385–397, 1963.
8. Kosaburo Hashiguchi. A decision procedure for the order of regular events. *Theoretical Computer Science*, 8 :69–72, 1979.
9. Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 24(2) :233–244, 1982.
10. Kosaburo Hashiguchi. Regular languages of star height one. *Information and Control*, 53(3) :199–210, 1982.
11. Kosaburo Hashiguchi. Representation theorems on regular languages. *J. Comput. Syst. Sci.*, 27(1) :101–115, 1983.
12. Kosaburo Hashiguchi. Relative star height, star height and finite automata with distance functions. In *Formal Properties of Finite Automata and Applications*, pages 74–88, 1988.
13. Daniel Kirsten. Desert automata and the finite substitution problem. In *STACS*, volume 2996 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2004.
14. Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO*, 3(39) :455–509, 2005.
15. Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3) :405–425, 1994.
16. Manfred Kufleitner. The height of factorization forests. In *MFCS*, volume 5162, pages 443–454, 2008.
17. Robert McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1-2) :167–176, 1967.
18. Imre Simon. Limited subsets of a free monoid. In *FOCS*, pages 143–150. IEEE, 1978.
19. Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72 :65–94, 1990.