Aix-Marseille Université

Faculté des Sciences

MÉMOIRE DE MASTER

En Informatique et Mathématiques Discrètes

Sujet :

Une approche algébrique à la minimisation des transducteurs à registres

(Algebraic approach to register minimisation in streaming string transducers)

Yahia Idriss BENALIOUA

Laboratoire d'Informatique et Systèmes

Encadrants : Nathan LHOTE et Pierre-Alain REYNIER

Année Universitaire : 2021 - 2022

Remerciements

Ce travail a été réalisé au Laboratoire d'Informatique et Systèmes à Aix-Marseille Université.

J'adresse mes sincères remerciements à mes encadrants, Messieurs Nathan Lhote et Pierre-Alain Reynier pour leur soutien et leurs encouragements, pour leur disponibilité et pour la confiance qu'ils ont placée en moi.

Je remercie les membres du jury pour le temps qu'ils ont consacré à examiner ce travail.

Je remercie également mes enseignants du M2 IMD pour tout ce qu'ils m'ont appris durant cette année.

Je remercie enfin ma famille et mes amis pour leur soutien indéfectible.

Résumé

L'étude des fonctions et relations entre les mots finis est une extension naturelle de la théorie des langages réguliers. Ces langages peuvent être décrits de différentes façons à l'aide d'automates, d'expressions régulières, d'algèbre, de logique... C'est pourquoi, faire bénéficier les fonctions de cette même diversité de caractérisations est l'un des objectifs de la communauté scientifique. Dans ce mémoire, nous nous intéressons à la classe des fonctions rationnelles. Cette classe est composée de fonctions pouvant être réalisées par des automates (unidirectionnels et non déterministes) munis de sorties, appelés transducteurs. Elle peut également être réalisée par une classe de transducteurs à registres (SST), et possède une caractérisation algébrique par un modèle appelé bimachine.

Les caractérisations algébriques des langages permettent de décider des sous-classes. Au cours de ce stage, nous avons cherché à résoudre des questions de simplification de modèles (comme la réduction du nombre de registres d'un SST) à l'aide de la représentation algébrique des fonctions rationnelles sous la forme de bimachines. Nous avons notamment étudié la sous-classe des fonctions multiséquentielles, qui sont des fonctions réalisables par une union finie de transducteurs déterministes. Nous avons tenté d'obtenir une caractérisation algébrique de cette classe à l'aide des bimachines. Nous nous sommes ensuite intéressés au nombre minimal de transducteurs nécessaires dans cette union et avons montré que ce nombre est exponentiel. Nous avons également étudié le problème de la minimisation de registres pour les SST et avons obtenu, à l'aide d'une équivalence avec les bimachines, un résultat pour une sous-classe de SST ainsi qu'un résultat partiel pour le cas général.

Table des matières

Introduction

Les langages réguliers peuvent être caractérisés à l'aide de plusieurs outils faisant intervenir différents domaines. Ces langages sont ceux pouvant être reconnus par des automates finis, ou pouvant être décrits par des expressions régulières, ou des formules de la logique MSO, ou dont la congruence syntaxique est d'indice fini... Chacune des caractérisations usuelles permet d'aborder la théorie des langages réguliers d'un angle différent, en fonction du problème posé, lui offrant ainsi une grande richesse.

Les automates peuvent être étendus en les munissant de sorties. Ce modèle, appelé *transducteur*, permet de réaliser des relations entre langages réguliers appelées *transductions* et, en particulier, des fonctions appelées *fonctions rationnelles*. Ces fonctions possèdent différentes caractérisations (cf. [14]) généralisant celles des langages réguliers.

Nous allons nous intéresser dans ce mémoire à leur caractérisation algébrique. Elle est basée sur un modèle appelé *bimachine* introduit par Schützenberger dans [19]. Toute fonction rationnelle peut être réalisée par une bimachine canonique qui lui est associée (cf. [13]), permettant notamment de décider des propriétés importantes de la fonction telles que son appartenance à une certaine sous-classe.

L'un de nos objectifs est alors d'utiliser cette caractérisation afin de décider si une fonction rationnelle donnée est *multiséquentielle*, i.e. réalisable par une union finie de transducteurs *séquentiels*, dont l'automate sous-jacent est déterministe. Cette question a déjà été résolue dans [16] en identifiant un motif présent dans tous les transducteurs réalisant des fonctions multiséquentielles. L'intérêt de notre approche est de réussir à décider la multiséquentialité indépendamment du modèle réalisant la fonction donnée. Nous avons également obtenu une borne exponentielle sur le nombre minimal de transducteurs de l'union, appelé *degré de séquentialité* de la fonction, en analysant la procédure de [16].

Les fonctions rationnelles peuvent également être réalisées par une classe de transducteurs à registres (Streaming String Transducers ou SST). Ce modèle, introduit dans [1], est constitué d'un automate déterministe muni d'un nombre fini de registres contenant des mots sur un alphabet de sortie. Ces registres sont mis à jour, à chaque transition, en fonction de la lettre lue, de l'état de l'automate et de leur valeur précédente et servent à définir la sortie à la fin de l'exécution. L'expressivité des SST varie en fonction du type de mises à jour autorisées. Les fonctions multiséquentielles, par exemple, sont réalisées par des SST où chaque registre ne dépend pas de la valeur des autres (cf. [10]).

Le problème naturel de minimisation pour les SST est celui de la minimisation de registres. Il s'agit de déterminer, étant donnée une fonction réalisée par un SST, le nombre minimal de registres nécessaires à sa réalisation. Pour les fonctions rationnelles, ce problème a été résolu dans [11], ici aussi, à l'aide d'un motif dans le transducteur réalisant la fonction. Nous proposons dans ce mémoire une solution pour une certaine sous classe de SST basée sur leur équivalence avec les bimachines. Nous avons également introduit un modèle alternatif de bimachines, correspondant

aux SST réalisant les fonctions rationnelles.

Nous commencerons par présenter, dans le premier chapitre, les transducteurs et les classes de fonctions qu'ils peuvent réaliser. Nous introduirons ensuite, dans le second, les caractérisations algébriques de ces fonctions et le modèle de bimachines. Le troisième chapitre sera quant à lui consacré aux transducteurs à registres. Enfin, la description de notre approche ainsi que des résultats que nous avons obtenus dans notre étude des fonctions multiséquentielles et du problème de la minimisation de registres feront l'objet des deux derniers chapitres.

Préliminaires et notations

Avant de commencer notre étude, introduisons quelques notions de base et notations dont nous nous servirons tout au long de ce mémoire.

Notations : Soient E et F deux ensembles et i et j deux entiers

- On notera $|E|$ le cardinal de E, $\mathcal{P}(E)$ son ensemble des parties et $\mathcal{P}_f(E)$ son ensemble des parties finies.
- On notera $E \times F$ le produit cartésien de E et F où π_E (resp. π_F) désignera la projection sur E (resp. F). Si $E = F$, ces projections seront notées respectivement π_1 et π_2.
- On notera $\mathcal{F}(E, F)$ l'ensemble des fonctions partielles de E vers F. Les relations binaires $R \subset E \times F$ telles que si xRy et xRz alors $y = z$ seront considérées comme des fonctions.
- Le domaine d'une fonction partielle f sera noté $dom(f)$ et la restriction de f à un sous-ensemble $D \subseteq dom(f)$ sera notée $f_{|D}$.
- $\mathbb{N} = \{0, 1, \dots\}$ désignera l'ensemble des entier naturels, $\mathbb{N}^* = \{1, 2, \dots\}$ désignera l'ensemble des entier positifs, et $[\![i, j]\!] = \{i, i+1, \dots, j\}$ désignera l'intervalle d'entiers entre i et j (avec $[\![i, j]\!] = \varnothing$ si $i > j$).

Mots et langages

Définitions: Considérons les définitions et notations suivantes :

- Un *alphabet* est un ensemble fini de *lettres*. Nous le noterons généralement Σ ou Γ.
- Un *mot* sur Σ est une suite finie de lettres $\sigma_1\sigma_2 \dots \sigma_n$ où $\sigma_i \in \Sigma, \forall i \in [\![1, n]\!]$.
- La *longueur* d'un mot w sera notée $|w|$. L'unique mot de longueur 0 est le *mot vide*. Il sera noté ϵ.
- Pour $n \in \mathbb{N}$, l'ensemble des mots de longueur n sur Σ sera noté Σ^n. L'ensemble des mots finis sur Σ sera noté Σ^* et $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$.
- Un ensemble de mots, soit un sous-ensemble de Σ^*, est appelé un *langage*.

On munit Σ^* d'une structure de monoïde :

Définition: Un *monoïde* $(M, \cdot, e)$ est un ensemble M muni d'une loi de composition interne $\cdot$ associative et possédant un élément neutre $e \in M$. $(M, \cdot, e)$ est un *groupe* si, de plus, tout élément de M admet un inverse pour la loi $\cdot$ i.e. $\forall m \in M, \exists m^{-1} \in M$ tel que $m \cdot m^{-1} = m^{-1} \cdot m = e$.

Lorsqu'il n'y a pas de confusion possible sur la loi, nous identifierons $(M, \cdot, e)$ et M.

La *concaténation* de deux mots $u = a_1 \dots a_n$ et $v = b_1 \dots b_m$ est le mot $a_1 \dots a_n b_1 \dots b_m$ que l'on notera $u \cdot v$ ou simplement uv. On a alors défini une loi de composition interne sur Σ^*

associative et d'élément neutre ϵ, lui conférant ainsi une structure de monoïde $(\Sigma^*, \cdot, \epsilon)$ appelé le *monoïde libre* sur Σ.

Soit $\Sigma^{-1} = \{\sigma^{-1} \,|\, \sigma \in \Sigma\}$ l'ensemble des inverses formels des lettres de Σ, i.e. Σ^{-1} est un ensemble en bijection avec Σ tel que $\Sigma \cap \Sigma^{-1} = \varnothing$ et tout mot $u \in (\Sigma \cup \Sigma^{-1})^*$ se réduit en une forme irréductible $\tilde{u}$ par les relations $\sigma \cdot \sigma^{-1} = \sigma^{-1} \cdot \sigma = \epsilon$ pour tout $\sigma \in \Sigma$. On munit alors l'ensemble des mots irréductibles de $(\Sigma \cup \Sigma^{-1})^*$ d'une structure de groupe, qu'on appellera le *groupe libre* sur Σ, en considérant pour loi interne $u \cdot v = \widetilde{uv}$.

Nous nous intéresserons aux langages *réguliers*. Ces langages possèdent plusieurs définitions équivalentes. Nous les définissons ici avec les automates et en donnons une caractérisation algébrique, cf. [6] ou [12] pour plus de détails.

Automates

Définition: Un *automate fini* sur un alphabet Σ est un 4-uplet $\mathcal{A} = (Q, I, F, \Delta)$ où :

— Q est un ensemble fini d'états.

— $I \subseteq Q$ (resp. $F \subseteq Q$) est l'ensemble des états initiaux (resp. finaux).

— $\Delta \subseteq Q \times \Sigma \times Q$ est un ensemble fini de transitions.

Notation : Les transitions $(p, \sigma, q) \in \Delta$ sont notées $p \xrightarrow{\sigma} q$.

Une *exécution* de $\mathcal{A}$ sur un mot $u = a_1 \dots a_n \in \Sigma^*$ est une suite finie d'états $(q_i)_{i \in [\![0,n]\!]}$ telle que pour tout $i \in [\![1, n]\!]$ il existe une transition $q_{i-1} \xrightarrow{a_i} q_i$. L'exécution est dite *acceptante* si $q_0 \in I$ et $q_n \in F$.

Le langage *reconnu* par $\mathcal{A}$, noté $[\![\mathcal{A}]\!]$, est l'ensemble de mots sur lesquels il existe une exécution acceptante de $\mathcal{A}$. On a alors la définition suivante :

Définition: Un langage $L \subseteq \Sigma^*$ est *régulier* s'il existe un automate $\mathcal{A}$ sur Σ tel que $[\![\mathcal{A}]\!] = L$.

Notation : Pour tous $p, q \in Q$ et $u \in \Sigma^*$, on notera $p \xrightarrow{u}_{\mathcal{A}} q$ s'il existe une exécution de $\mathcal{A}$ sur u débutant en p et se terminant en q.

Définitions: Soit $\mathcal{A} = (Q, I, F, \Delta)$ un automate et $q \in Q$

— q est dit *accessible* s'il existe $p \in I$ et $u \in \Sigma^*$ tels que $p \xrightarrow{u}_{\mathcal{A}} q$. Symétriquement, q est dit *co-accessible* s'il existe $p \in F$ et $u \in \Sigma^*$ tels que $q \xrightarrow{u}_{\mathcal{A}} p$.

— Si tous ses états sont (co-)accessibles, $\mathcal{A}$ est dit *(co-)accessible*. On dit que $\mathcal{A}$ est *émondé* s'il est à la fois accessible et co-accessible.

— On dit que $\mathcal{A}$ est *ambigu* s'il existe un mot sur lequel $\mathcal{A}$ possède deux exécutions acceptantes différentes.

— $\mathcal{A}$ est dit *complet* si pour tous $p \in Q$ et $\sigma \in \Sigma$ il existe $q \in Q$ tel que $(p, \sigma, q) \in \Delta$.

On considérera par la suite des automates supposés complets.

Définition: $\mathcal{A}$ est dit *déterministe* (resp. co-déterministe) si :

— $\mathcal{A}$ possède un unique état initial q_0 (resp. final q_f) i.e. $|I| = 1$ (resp. $|F| = 1$).

— Si (p, a, q) et $(p, a, q') \in \Delta$ alors $q = q'$ (resp. si (p, a, q) et $(p', a, q) \in \Delta$ alors $p = p'$).

On définit alors une fonction partielle $- \cdot_{\mathcal{A}} - : Q \times \Sigma \to Q$ (resp. $- \cdot_{\mathcal{A}} - : \Sigma \times Q \to Q$) par $p \cdot_{\mathcal{A}} a = q$ si $\exists q \in Q$ (resp. $a \cdot_{\mathcal{A}} q = p$ si $\exists p \in Q$) tel que $(p, a, q) \in \Delta$.

Si $\mathcal{A}$ est complet, on étend cette fonction en une action à droite du monoïde Σ^* sur Q $- \cdot - : Q \times \Sigma^* \to Q$ (resp. à gauche $- \cdot - : \Sigma^* \times Q \to Q$) en posant $q \cdot \epsilon = q$ et $q \cdot (au) = (q \cdot_{\mathcal{A}} a) \cdot u$ (resp. $\epsilon \cdot q = q$ et $(ua) \cdot q = u \cdot (a \cdot_{\mathcal{A}} q)$) pour tous $q \in Q$, $a \in \Sigma$ et $u \in \Sigma^*$.

Proposition *Tout automate est équivalent à (i.e. reconnaît le même langage que) un automate déterministe et complet.*

Un langage est donc régulier si et seulement s'il est reconnu par un automate déterministe et complet. La section suivante présente une caractérisation de ces langages à l'aide de congruences.

Congruences et automate minimal

Définition: Une *relation d'équivalence* $\sim$ sur Σ^* est une relation binaire réflexive, symétrique et transitive. Si, de plus, pour tous $u, v \in \Sigma^*$ et $\sigma \in \Sigma$, si $u \sim v$ alors $u\sigma \sim v\sigma$ (resp. $\sigma u \sim \sigma v$), on dit que $\sim$ est une *congruence droite* (resp. *gauche*).

Le nombre de classes d'équivalences de $\sim$, soit $|\Sigma^*/\sim|$, est appelé son *indice*.

Notation : Pour $u \in \Sigma^*$, on note $[u]_\sim$ (ou simplement $[u]$) la classe d'équivalence de u pour $\sim$.

On associe à tout automate $\mathcal{A}$ déterministe (resp. co-déterministe) d'état initial q_0 (resp. final q_f) la congruence droite (resp. gauche) $\sim_{\mathcal{A}}$ définie pour tous $u, v \in \Sigma^*$ par $u \sim_{\mathcal{A}} v \Leftrightarrow q_0 \cdot u = q_0 \cdot v$ (resp. $u \cdot q_f = v \cdot q_f$). On peut alors identifier toute classe d'équivalence de $\sim_{\mathcal{A}}$, que nous noterons simplement $[u]_{\mathcal{A}}$ ou $[u]$, à l'état $q_0 \cdot u$ (resp. $u \cdot q_f$) atteint par $\mathcal{A}$ en lisant les mots de $[u]_{\mathcal{A}}$. On notera alors pour tout état $q \in Q$, $[q] \in \Sigma^*/\sim_{\mathcal{A}}$ pour désigner la classe d'équivalence à laquelle il est identifié.

Définition: Si $\sim_1$ et $\sim_2$ sont deux relations d'équivalence sur Σ^*, on dit que $\sim_1$ est *plus fine* que $\sim_2$ (ou $\sim_2$ est *plus grossière* que $\sim_1$) si, pour tous $u, v \in \Sigma^*$, si $u \sim_1 v$ alors $u \sim_2 v$. On note alors $\sim_1 \sqsubseteq \sim_2$.

En particulier, on dit qu'un automate $\mathcal{A}_1$ est *plus fin* qu'un automate $\mathcal{A}_2$ si $\sim_{\mathcal{A}_1} \sqsubseteq \sim_{\mathcal{A}_2}$ et on note simplement $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$. Dans ce cas, $\mathcal{A}_2$ possède un nombre d'états inférieur à celui de $\mathcal{A}_1$.

L'*automate minimal* d'un langage L peut alors être défini comme l'automate déterministe le moins fin reconnaissant L.

Définition: Pour tout langage $L \subseteq \Sigma^*$ et $u \in \Sigma^*$, on définit le *résiduel* (à gauche) de L par rapport à u par $u^{-1}L = \{v \in \Sigma^* \,|\, uv \in L\}$. On peut alors définir la *congruence syntaxique* (droite) associée à L, que nous noterons $\sim_L$, pour tous $u, v \in \Sigma^*$ par $u \sim_L v \Leftrightarrow u^{-1}L = v^{-1}L$ (i.e. $\forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$).

Cette congruence, aussi appelée la congruence de Myhill-Nerode, nous donne une caractérisation algébrique des langages réguliers.

Théorème (Myhill-Nerode)
Un langage L est régulier si et seulement si $\sim_L$ est d'indice fini.

De plus, si cette congruence est d'indice fini, elle permet de définir l'automate minimal $\mathcal{A}_L = (Q_L, I_L, F_L, \Delta_L)$ de L par :

$$Q_L = \Sigma^*/\sim_L,\ I_L = \{[\epsilon]\},\ F_L = \{[u] \,|\, u \in L\} \text{ et } \Delta_L = \{([u], \sigma, [u\sigma]) \,|\, u \in \Sigma^*, \sigma \in \Sigma\}$$

(Prendre des résiduels à droite donne, symétriquement, une congruence syntaxique gauche donnant un automate co-déterministe minimal)

Chapitre 1

Transductions

Les transducteurs sont des automates munis de sorties leur permettant de réaliser des relations, appelées transductions, entre langages réguliers. Plusieurs classes de fonctions peuvent être définies selon les propriétés des transducteurs pouvant les réaliser. Dans ce chapitre, nous introduisons ce modèle et présentons les classes de fonctions qui interviendront dans les chapitres suivants.

1.1 Transducteurs

Définition 1.1: Un *transducteur*, d'alphabet d'entrée Σ (resp. de sortie Γ), est un 4-uplet $\mathcal{T} = (\mathcal{A}, s, i, t)$ où :

- $\mathcal{A} = (Q, I, F, \Delta)$ est un automate fini sur Σ.
- $s : \Delta \to \Gamma^*$ est la fonction de sortie.
- $i : I \to \Gamma^*$ est la fonction de sortie initiale.
- $t : F \to \Gamma^*$ est la fonction de sortie terminale.

Notation : Les transitions $\delta = (p, \sigma, q) \in \Delta$ avec $s(\delta) = w$ sont notées $p \xrightarrow{\sigma|w} q$.

Comme pour les automates, une *exécution* de $\mathcal{T}$ sur un mot $u = a_1 \dots a_n \in \Sigma^*$ est une suite finie d'états $(q_i)_{i \in [\![0,n]\!]}$ telle que pour tout $i \in [\![1, n]\!]$ il existe une transition $q_{i-1} \xrightarrow{a_i|w_i} q_i$. L'exécution est *acceptante* si $q_0 \in I$ et $q_n \in F$.

La sortie d'une telle exécution est le mot $i(q_0)w_1 \dots w_n t(q_n) \in \Gamma^*$, résultant de la concaténation des mots produits à chaque transition et du mot donné par l'image de l'état initial (resp. final) par la fonction de sortie initiale (resp. terminale).

La *transduction réalisée* par $\mathcal{T}$ est la relation $[\![\mathcal{T}]\!] \subset \Sigma^* \times \Gamma^*$ définie par $(u, w) \in [\![\mathcal{T}]\!]$ si et seulement s'il existe une exécution acceptante de $\mathcal{T}$ sur u ayant pour sortie w.

Nous pouvons supposer, sans perte de généralité, que $\Sigma = \Gamma$.

On dira qu'un transducteur est *fonctionnel* si la relation qu'il réalise est une fonction. Il est possible de décider en temps polynomial (cf. [4]) si un transducteur est fonctionnel. On ne considérera dans ce mémoire que des transducteurs fonctionnels.

1.2 Classes de transductions

1.2.1 Fonctions rationnelles

Définition 1.2: Une fonction $f : \Sigma^* \to \Sigma^*$ est *rationnelle* s'il existe un transducteur $\mathcal{T}$ sur Σ tel que $[\![\mathcal{T}]\!] = f$.

Exemple 1.1: Soient $\Sigma = \{a, b\}$, $\sigma \in \Sigma$ et $w \in \Sigma^*$.

On définit la fonction *last* sur Σ^+ par $last(w\sigma) = \sigma^{|w|+1}$. Elle transforme toutes les lettres d'un mot en sa dernière lettre.

Cette fonction est rationnelle. Elle est réalisée par le transducteur représenté par la figure ci-contre. Il possède deux états initiaux : q_a et q_b lui permettant de "prédire" de manière non déterministe la dernière lettre de son mot d'entrée et de produire les sorties adéquates. Son unique état final est q_f. Sur la figure, les valeurs étiquetant les flèches entrantes dans les états initiaux (resp. sortante des états finaux) sont celles de la fonction de sortie initiale (resp. terminale). Ici, ces deux fonctions sont toujours égales à ϵ (dans ce cas les étiquettes seront omises par la suite).

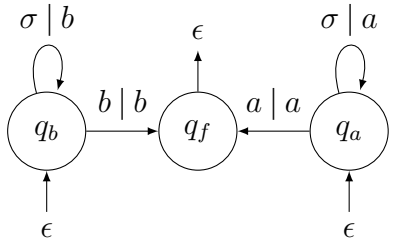


FIGURE 1.1 – Transducteur réalisant *last* ($\sigma \in \Sigma$)

Exemple 1.2: La fonction *mir* transformant tout mot en son image miroir (i.e. $mir(a_1 \dots a_n) = a_n \dots a_1$) n'est pas une fonction rationnelle. Intuitivement, cela vient du fait qu'un transducteur ne peut lire son entrée qu'une seule fois de gauche à droite. Il ne peut donc pas produire de sortie avant d'avoir atteint la fin du mot et doit "retenir" un nombre non borné de lettres.

Cette fonction est dite *régulière*. Elle peut être réalisée par un transducteur bidirectionnel pouvant changer de sens de lecture pendant son exécution (cf. [14]).

Nous verrons, au chapitre suivant, une caractérisation algébrique des fonctions rationnelles basée sur un modèle appelé *bimachine*.

1.2.2 Fonctions séquentielles

Une sous-classe de la classe des fonctions rationnelles, possédant des propriétés intéressantes, est la classe des fonctions séquentielles.

Définition 1.3: Une fonction $f : \Sigma^* \to \Sigma^*$ est *séquentielle* s'il existe un transducteur $\mathcal{T} = (\mathcal{A}, s, i, t)$ sur Σ tel que $[\![\mathcal{T}]\!] = f$ et $\mathcal{A}$ est un automate déterministe.

On dira alors d'un transducteur dont l'automate sous-jacent est (co-)déterministe qu'il est *(co-)séquentiel*.

Exemple 1.3: Le transducteur de la figure 1.1 est co-séquentiel. Il est cependant impossible de construire un transducteur séquentiel réalisant la fonction *last*. Cela vient du fait qu'un automate déterministe ne peut pas "deviner" la dernière lettre de son mot d'entrée.

Étant donnée une fonction rationnelle, il est possible de décider en temps polynomial si elle est séquentielle (cf. [4]). Il existe également une caractérisation algébrique de la classe des fonctions séquentielles, similaire au théorème de Myhill-Nerode, permettant d'obtenir le transducteur minimal. Nous présenterons cette caractérisation au chapitre suivant.

Exemple 1.4: Soit $\Sigma = \{a, b\}$. Pour $\sigma \in \Sigma$, soit $\overline{\sigma} = \begin{cases} a & \text{si } \sigma = b \\ b & \text{si } \sigma = a \end{cases}$

La fonction inversant toutes les lettres d'un mot (i.e. $\overline{a_1 a_2 \dots a_n} = \overline{a_1}\,\overline{a_2} \dots \overline{a_n}$) est une fonction séquentielle réalisée par le transducteur séquentiel ci-contre.

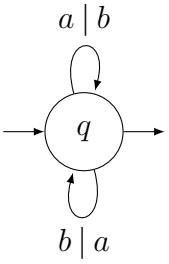


FIGURE 1.2 – Transducteur réalisant $\overline{\cdot}$

1.2.3 Fonctions multiséquentielles

Nous allons nous intéresser à une classe intermédiaire entre celle des fonctions séquentielles et celle des fonctions rationnelles. Introduites dans [9] sous le nom de fonctions "pluri-sous-séquentielles", les fonctions *multiséquentielles* sont des fonctions rationnelles pouvant être décomposées en une union finie de fonctions séquentielles.

Définition 1.4: Une fonction $f : \Sigma^* \to \Sigma^*$ est *multiséquentielle* s'il existe un nombre fini k de fonctions séquentielles $f_1, \dots, f_k$ telles que $f = \bigcup_{i=1}^{k} f_i$.

Exemple 1.5: La fonction *last* définie précédemment est multiséquentielle.

En effet, il suffit de décomposer son domaine suivant la dernière lettre du mot d'entrée. La fonction peut alors être réalisée séquentiellement sur chacune des deux parties comme le montre la figure ci-contre.

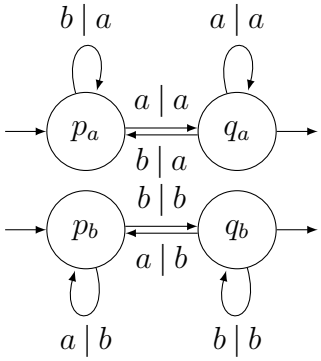


FIGURE 1.3 – Transducteur multiséquentiel réalisant *last*

Définition 1.5: Le plus petit entier k tel qu'une fonction multiséquentielle f s'écrit comme réunion de k fonctions séquentielles est appelé son *degré de séquentialité*. On dit alors que f est k-séquentielle.

La fonction *last* de notre exemple est 2-séquentielle.

Certaines fonctions rationnelles ne sont pas multiséquentielles :

Exemple 1.6: Soit $\Sigma = \{a, b\}$ et $\Sigma_\# = \Sigma \cup \{\#\}$.

Considérons une version itérée de la fonction *last* opérant sur des mots décomposés en un nombre de blocs prédéfini : pour tout entier n on définit la fonction $last^n$ sur $(\Sigma^+\#)^{n-1}\Sigma^+$ pour tous $w_1, \dots, w_n \in \Sigma^+$ par

$$last^n(w_1 \# w_2 \# \dots \# w_n) = last(w_1) \# last(w_2) \# \dots \# last(w_n)$$

Nous démontrerons plus tard que cette fonction est 2^n-séquentielle.

La fonction $last^*$ opérant sur un nombre non borné de blocs (i.e. $last^* = \bigcup_{n \in \mathbb{N}^*} last^n$) n'est pas multiséquentielle. Pourtant, c'est une fonction rationnelle. En effet, il suffit d'ajouter les transitions $q_f \xrightarrow{\#|\#} q_a$ et $q_f \xrightarrow{\#|\#} q_b$ au transducteur de la figure 1.1 afin d'obtenir un transducteur réalisant $last^*$.

Il est démontré dans [16] qu'il est possible de décider, en temps polynomial, si une fonction rationnelle est multiséquentielle. La caractérisation donnée dans [16] (ou [9]) consiste à déterminer si un certain motif apparaît dans le transducteur définissant la fonction. Cependant, il n'existe pour l'instant aucune caractérisation algébrique de la classe. L'un des objectifs de ce stage était alors d'essayer d'obtenir une propriété, caractérisant les fonctions multiséquentielles, indépendante de la machine réalisant la fonction considérée.

Chapitre 2

Caractérisations algébriques

Intéressons nous maintenant aux caractérisations algébriques des classes de fonctions que nous avons introduites au chapitre précédent. Ces caractérisations permettent, entre autre, d'obtenir un objet canonique, similaire à l'automate minimal dans le cas des langages réguliers, réalisant la fonction donnée.

2.1 Fonctions séquentielles

Le cas le plus simple est celui des fonctions séquentielles. Comme pour le théorème de Myhill-Nerode, une congruence syntaxique caractérisant ces fonctions a été introduite dans [7]. Afin de la définir, nous aurons besoin des notations suivantes :

Notations : Soit Σ un alphabet et u et v deux mots sur Σ

— On note $u \preceq v$ si u est un préfixe de v (i.e. s'il existe $w \in \Sigma^*$ tel que $v = uw$).

— On note $u \wedge v$ le plus long préfixe commun de u et v (i.e. $u \wedge v \preceq u, v$ et $\forall w \preceq u, v$ on a $w \preceq u \wedge v$). Cette notation se généralise à tout langage L. On note $\bigwedge L$ le plus long préfixe commun de tous les mots de L avec $\bigwedge \varnothing = \epsilon$.

Définition 2.1: Soit $f : \Sigma^* \to \Sigma^*$ une fonction. On définit la fonction $\widehat{f} : \Sigma^* \to \Sigma^*$ par

$$\widehat{f}(u) = \bigwedge \left\{ f(uw) \,\middle|\, w \in u^{-1}dom(f) \right\} = \bigwedge_{uw \in dom(f)} f(uw)$$

On définit alors la *congruence syntaxique* associée à f, notée $\sim_f$, par

$$u \sim_f v \Leftrightarrow \begin{cases} u^{-1}dom(f) = v^{-1}dom(f) \\ \forall w \in u^{-1}dom(f),\ \widehat{f}(u)^{-1} f(uw) = \widehat{f}(v)^{-1} f(vw) \end{cases}$$

La première condition est la même que pour la congruence de Myhill-Nerode, elle assure que le domaine de la fonction est reconnu. La seconde, quant à elle, assure que les sorties produites après la lecture de deux mots équivalents sont les mêmes.

On a alors, comme dans le cas des langages réguliers :

Théorème 2.1 (Choffrut [7])
Une fonction f est séquentielle si et seulement si $\sim_f$ est d'indice fini.

Dans ce cas, cette congruence permet de définir le transducteur minimal de f (i.e. le transducteur séquentiel le moins fin réalisant f) $\mathcal{T}_f = (\mathcal{A}_f, s_f, i_f, t_f)$ où $\mathcal{A}_f = (Q_f, I_f, F_f, \Delta_f)$ est défini, comme l'automate minimal dans le cas des langages réguliers, par :

$Q_f = \Sigma^*/\sim_f$, $I_f = \{[\epsilon]\}$, $F_f = \{[u] \,|\, u \in dom(f)\}$ et $\Delta_f = \{([u], \sigma, [u\sigma]) \,|\, u \in \Sigma^*, \sigma \in \Sigma\}$

Pour les sorties :

— On définit s_f, pour toute transition $\delta = ([u], \sigma, [u\sigma]) \in \Delta_f$, $s_f(\delta) = \widehat{f}(u)^{-1}\widehat{f}(u\sigma)$.

— On définit i_f par $i_f([\epsilon]) = \widehat{f}(\epsilon)$ et t_f, pour tout $u \in dom(f)$, par $t_f([u]) = \widehat{f}(u)^{-1} f(u)$.

2.2 Fonctions rationnelles

La caractérisation algébrique des fonctions rationnelles passe par un modèle appelé *bimachine*. Ce modèle, introduit par Schützenberger dans [19], possède la même expressivité que les transducteurs.

2.2.1 Bimachines

Comme son nom l'indique, une *bimachine* est composée de deux automates et d'une fonction de sortie dépendant de leurs états et des lettres lues. Ces deux automates lui permettent de traiter son entrée dans les deux sens simultanément (de gauche à droite et inversement).

Définition 2.2: Une *bimachine*, d'alphabet Σ, est un 5-uplet $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ où :

— $\mathcal{L} = (L, \{l_0\}, F_\mathcal{L}, \Delta_\mathcal{L})$ est un automate fini déterministe appelé son *automate droit*.

— $\mathcal{R} = (R, I_\mathcal{R}, \{r_0\}, \Delta_\mathcal{R})$ est un automate fini co-déterministe appelé son *automate gauche*.

— $\omega : L \times \Sigma \times R \to \Sigma^*$ est une fonction partielle de sortie.

— $\lambda : I_\mathcal{R} \to \Sigma^*$ et $\rho : F_\mathcal{L} \to \Sigma^*$ sont, respectivement, la fonction de sortie terminale gauche et droite.

On étend ω en une fonction partielle $\omega : L \times \Sigma^* \times R \to \Sigma^*$ en posant $\omega(l, \epsilon, r) = \epsilon$ et $\omega(l, uv, r) = \omega(l, u, v \cdot r)\,\omega(l \cdot u, v, r)$ pour tous $l \in L$, $r \in R$ et $u, v \in \Sigma^*$.

La bimachine réalise la fonction $[\![\mathcal{B}]\!] : \Sigma^* \to \Sigma^*$ définie par

$$[\![\mathcal{B}]\!](u) = \lambda(u \cdot r_0)\,\omega(l_0, u, r_0)\,\rho(l_0 \cdot u)$$

Ou, plus explicitement, si $u = a_1 \dots a_n$

$$\begin{aligned}[\![\mathcal{B}]\!](u) =& \lambda(u \cdot r_0)\,\omega(l_0,\, a_1\,, a_2 \dots a_n \cdot r_0)\,\omega(l_0 \cdot a_1,\, a_2\,, a_3 \dots a_n \cdot r_0) \dots \\ & \dots \omega(l_0 \cdot a_1 \dots a_{n-2},\, a_{n-1}\,, a_n \cdot r_0)\,\omega(l_0 \cdot a_1 \dots a_{n-1},\, a_n\,, r_0)\,\rho(l_0 \cdot u)\end{aligned}$$

Exemple 2.1: Soit $\Sigma = \{a, b\}$. On définit la fonction *swap*, inversant la première et la dernière d'un mot. i.e. pour tout $w \in \Sigma^*$, $swap(w) = w$ si $|w| < 2$ et pour tous $\sigma, \tau \in \Sigma$, $swap(\sigma w \tau) = \tau w \sigma$.

C'est une fonction rationnelle, réalisée par la bimachine de la figure 2.1, où les sorties sont définies, pour tous $\sigma \in \Sigma, l \in L$ et $r \in R$, par $\omega(l, \sigma, r) = \begin{cases} \tau & \text{si } \begin{array}{r} l = l_0 \text{ et } r = r_\tau \\ \text{ou } l = l_\tau \text{ et } r = r_0 \end{array} \text{ où } \tau \in \Sigma \\ \sigma & \text{sinon} \end{cases}$ et $\lambda(r) = \rho(l) = \epsilon$.

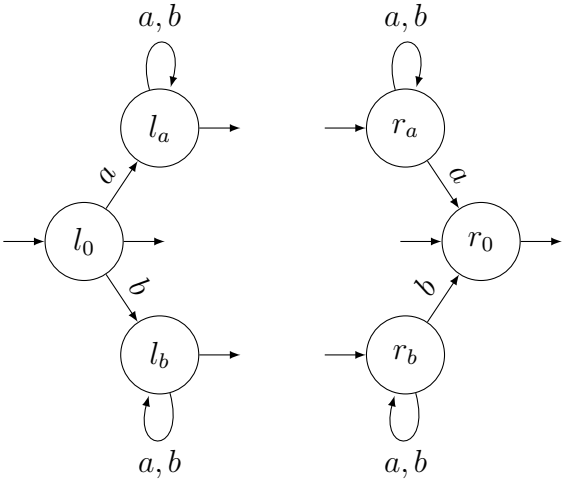


FIGURE 2.1 – Bimachine réalisant *swap*

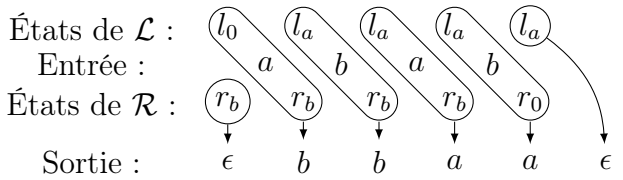


FIGURE 2.2 – Exemple d'exécution de la bimachine sur le mot *abab*

2.2.2 Congruences et bimachines minimales

Dans [18], les auteurs introduisent une congruence gauche canonique associée à toute fonction rationnelle, permettant d'obtenir l'automate droit minimal de cette fonction. i.e. l'automate droit le moins fin de tous les automates droits des bimachines réalisant la fonction.

Cette congruence est définie à l'aide de la *distance préfixe*, notée $\|\cdot,\cdot\|$, définie pour tous mots u et v sur un alphabet Σ par $\|u,v\| = |u| + |v| - 2|u \wedge v| = |u^{-1}v|$.

Définition 2.3: Soit $f : \Sigma^* \to \Sigma^*$ une fonction. On définit la *congruence syntaxique gauche* associée à f, notée $\leftsquigarrow_f$, par

$$u \leftsquigarrow_f v \Leftrightarrow \begin{cases} dom(f)u^{-1} = dom(f)v^{-1} \\ \sup\limits_{w \in dom(f)u^{-1}} \|f(wu), f(wv)\| < \infty \end{cases}$$

Si f est rationnelle, $\leftsquigarrow_f$ est d'indice fini et on construit naturellement l'automate droit minimal de f $\mathcal{R}_f = (R_f, I_{\mathcal{R}_f}, F_{\mathcal{R}_f}, \Delta_{\mathcal{R}_f})$ en prenant $R_f = \Sigma^*/\sim_f$, $I_{\mathcal{R}_f} = \{[u] \,|\, u \in dom(f)\}$, $F_{\mathcal{R}_f} = \{[\epsilon]\}$, et $\Delta_{\mathcal{R}_f} = \{([\sigma u], \sigma, [u]) \,|\, u \in \Sigma^*, \sigma \in \Sigma\}$. La réciproque nécessite cependant que la préimage par f de tout langage régulier est un langage régulier (cf. [18]).

Il est, bien sûr, possible de définir symétriquement une congruence syntaxique droite $\rightsquigarrow_f$ associée à f permettant d'obtenir l'automate gauche minimal.

Nous pouvons obtenir, à partir de cet automate droit, une bimachine canonique réalisant la fonction. En effet, une méthode permettant de construire l'automate gauche minimal associé à l'automate droit d'une bimachine réalisant une fonction rationnelle est décrite dans [13]. Cet automate est construit à l'aide d'une congruence similaire à celle de la définition 2.1 où l'information donnée par l'automate droit est prise en compte. Cette construction permet aussi de réduire l'automate droit d'une bimachine dont l'automate gauche est fixé, permettant ainsi, en réduisant l'automate droit par rapport à l'automate gauche puis inversement, d'obtenir une bimachine minimale (i.e. telle que tout autre bimachine réalisant la fonction a un automate droit ou gauche plus fin).

Chapitre 3

Transducteurs à registres

Un modèle alternatif de transducteurs a été introduit dans [1]. Ce modèle appelé *transducteur à registres* (Streaming String Transducer ou SST) est constitué d'un automate déterministe muni d'un nombre fini de registres contenant des mots sur l'alphabet de sortie. Ces registres sont mis à jour à chaque transition, en fonction de leurs valeurs précédentes, par des opérations de concaténation et servent à définir la sortie à la fin de l'exécution.

L'expressivité des transducteurs à registres ne se limite pas qu'aux fonctions rationnelles. En effet, la figure ci-contre représente un transducteur à registres réalisant la fonction *mir* de l'exemple 1.2. L'unique registre X de ce SST est initialisé à ϵ (dans ce cas les étiquettes seront omises par la suite) puis mis à jour en ajoutant à gauche de sa valeur actuelle la lettre lue à chaque transition. La sortie est alors la valeur de X à la fin de l'exécution (l'image miroir du mot d'entrée).

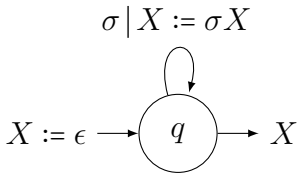


FIGURE 3.1 – SST réalisant *mir* ($\sigma \in \Sigma$)

En imposant des restrictions sur les opérations autorisées pour la mise à jour des registres, il est possible de définir des transducteurs à registres réalisant uniquement les fonctions des classes de transductions que nous avons introduites précédemment.

3.1 Classes de fonctions et transducteurs à registres

Les transducteurs à registres réalisant les fonctions rationnelles sont ceux dont toutes les mises à jour sont de la forme $X := Yw$ où X et Y sont deux registres (non nécessairement distincts) et w est un mot sur l'alphabet de sortie. On ne considérera dans ce mémoire que les SST de cette classe. Ils sont définis formellement comme suit :

Définition 3.1: Un *transducteur à registres*, d'alphabet Σ, est un 5-uplet $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ où :

- $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$ est un automate fini déterministe.
- $\mathcal{X}$ est un ensemble fini de registres.
- $\mathfrak{U} : \Delta \to \mathcal{F}(\mathcal{X}, \mathcal{X} \times \Sigma^*)$ est la fonction (totale) de mise à jour des registres.
- $v_0 : \mathcal{X} \to \Sigma^*$ est la fonction d'initialisation des registres.
- $\mu : F \to \mathcal{X} \times \Sigma^*$ est la fonction (totale) de sortie.

On considère les fonctions $\begin{array}{l} \mathfrak{f} : \Delta \to \mathcal{F}(\mathcal{X}, \mathcal{X}) \\ \delta \mapsto \mathfrak{f}_\delta = \pi_{\mathcal{X}} \circ \mathfrak{U}(\delta) \end{array}$ et $\begin{array}{l} \mathfrak{a} : \Delta \to \mathcal{F}(\mathcal{X}, \Sigma^*) \\ \delta \mapsto \mathfrak{a}_\delta = \pi_{\Sigma^*} \circ \mathfrak{U}(\delta) \end{array}$.

La fonction $\mathfrak{f}_\delta$ sera appelée le *flow* de la transition δ.

Les configurations d'un SST sont des couples $(q, v) \in Q \times \mathcal{F}(\mathcal{X}, \Sigma^*)$. On dit que v est une *valuation*. Elle associe à chaque registre le mot qu'il contient.

Une *exécution* de $\mathcal{T}$ sur un mot $u = a_1 \dots a_n \in \Sigma^*$ est une suite finie de configurations $(q_i, v_i)_{i \in [\![0,n]\!]}$ telle que, pour tout $i \in [\![1,n]\!]$, $\delta_i = (q_{i-1}, a_i, q_i) \in \Delta$ et $v_i(X) = v_{i-1}(\mathfrak{f}_{\delta_i}(X))\mathfrak{a}_{\delta_i}(X)$ pour tout registre $X \in dom(v_i) = \mathfrak{f}_{\delta_i}^{-1}(dom(v_{i-1}))$.

L'exécution est *acceptante* si $q_n \in F$, $\pi_{\mathcal{X}} \circ \mu(q_n) \in dom(v_n)$, et le premier terme de la suite est (q_0, v_0) où q_0 est l'état initial de $\mathcal{A}$ et v_0 est la fonction d'initialisation des registres.

$\mathcal{T}$ réalise une fonction $[\![\mathcal{T}]\!] : \Sigma^* \to \Sigma^*$ définie, sur les mots $u \in \Sigma^*$ tels qu'il existe une exécution acceptante $(q_i, v_i)_{i \in [\![0,n]\!]}$ de $\mathcal{T}$ sur u, par

$$[\![\mathcal{T}]\!](u) = v_n(\pi_{\mathcal{X}} \circ \mu(q_n))\, \pi_{\Sigma^*} \circ \mu(q_n)$$

Remarque : Cette définition est légèrement différente de la définition classique des SST. Usuellement, les fonctions de valuation et de mises à jour des registres sont des fonctions totales. Tous les registres doivent avoir une valeur tout au long de l'exécution. Notre définition permet d'éviter des problèmes liés aux domaines lors de la transformation des bimachines en SST que nous verrons au cinquième chapitre.

On dit qu'un SST est *à registres indépendants* si le flow de toute transition est égal à l'identité (i.e. si toutes les mises à jour sont de la forme $X := Xw$). Il est démontré dans [10] que ces SST correspondent à ceux réalisant les fonctions multiséquentielles :

Proposition 3.1 *Pour tout entier k, une fonction est k-séquentielle si et seulement si elle peut être réalisée par un SST possédant k registres indépendants.*

En particulier, une fonction est séquentielle si et seulement si elle est réalisée par un SST possédant un unique registre.

3.2 Minimisation de registres

Le problème naturel de minimisation pour les SST est celui de la minimisation de registres. Il s'énonce comme suit : étant donné un SST $\mathcal{T}$ et un nombre k, existe-t-il un SST $\mathcal{T}'$ équivalent à $\mathcal{T}$ (i.e. $[\![\mathcal{T}']\!] = [\![\mathcal{T}]\!]$) possédant k registres.

Dans le cas des fonctions rationnelles, ce problème a été résolu dans [11] par une méthode consistant à déterminer si un certain motif apparaît dans un transducteur réalisant la fonction donnée. Une méthode similaire a été utilisée dans [10] pour les SST à registres indépendants. Dans ce cas, résoudre ce problème revient à déterminer le degré de séquentialité de la fonction donnée.

Dans cette étude, nous avons tenté de résoudre ce problème à l'aide des bimachines. Cette approche algébrique vise à répondre à ces questions en identifiant les propriétés intrinsèques aux fonctions, indépendantes du modèle utilisé pour les représenter, décidant de leur réalisabilité par un modèle donné. Les résultats que nous avons obtenus feront l'objet des chapitres suivants.

Chapitre 4

Fonctions multiséquentielles

Comme nous l'avons souligné dans la section 1.2.3, les fonctions multiséquentielles ne possèdent pas encore de caractérisation algébrique. Nous avons alors tenté d'en obtenir une à partir de celle des fonctions rationnelles. Ce chapitre décrit notre approche. Notre étude nous a aussi mené à analyser l'algorithme de déterminisation faible présenté dans [16] afin d'obtenir une borne sur le degré de séquentialité.

4.1 Caractérisation algébrique

L'automate droit d'une bimachine peut être vu comme un automate fournissant des informations sur la suite du mot (look-ahead régulier) à l'automate gauche pour lui permettre de réaliser séquentiellement la fonction. Il existe alors un lien entre les états initiaux de l'automate droit de la bimachine réalisant une fonction multiséquentielle et sa décomposition en union de fonctions séquentielles.

La fonction *swap* de l'exemple 2.1 est une fonction 2-séquentielle. Les mots de son domaine peuvent être séparés selon leur dernière lettre et elle peut ainsi être réalisée séquentiellement sur chacune des deux parties. Nous pouvons remarquer que le rôle de son automate droit minimal (figure 2.1) est, justement, de distinguer les mots se terminant par a ou b.

Plusieurs exemples, dont notamment toutes les fonctions multiséquentielles citées jusqu'à présent dans ce mémoire, semblaient confirmer que restreindre la fonction aux classes d'équivalences correspondant aux états initiaux de son automate droit minimal permettait, dans le cas des fonctions multiséquentielles, d'obtenir une décomposition en union finie de fonctions séquentielles.

Cette classification ne fonctionne évidemment pas pour les fonctions qui ne sont pas multiséquentielles. La figure ci-contre représente l'automate droit minimal de la fonction (non multiséquentielle) $last^*$ de l'exemple 1.6. Si l'on prend par exemple l'état initial r_a, il est facile de montrer que $last^*$ restreinte à $[r_a]$ n'est pas séquentielle. En effet, d'après [8], une fonction est séquentielle si et seulement si elle est à *variations bornées*.

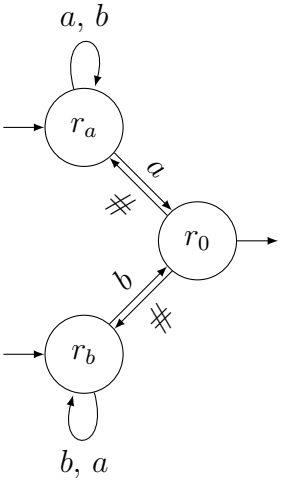


FIGURE 4.1 – $\mathcal{R}_{last^*}$

Définition 4.1: Une fonction $f : \Sigma^* \to \Sigma^*$ est à *variations bornées* si $\forall l \geq 0, \exists L \geq 0$ tel que si $u, v \in dom(f)$ et $\|u, v\| \leq l$ alors $\|f(u), f(v)\| \leq L$.

Il suffit alors de prendre, pour $l = 1$ et pour tout $L \geq 0$, les mots $u = a\#a^{L/2}$ et $v = ub$. On a bien $u, v \in [r_a]$ et $\|u, v\| = 1 \leq l$ mais $\|last^*(u), last^*(v)\| = \|a\#a^{L/2}, a\#b^{L/2}b\| = L + 1 > L$. D'où la non séquentialité de $last^*_{|[r_a]}$.

Nous avons cependant réussi à construire un exemple de fonction multiséquentielle ne vérifiant pas cette propriété, en prenant une version totale de $last^n$ appliquant la fonction $last$ sur les n premiers blocs de son entrée et ignorant le reste. En considérant que $last(\epsilon) = \epsilon$, on définit, pour tout entier n, la fonction totale $last^n_t : \Sigma^*_\# \to \Sigma^*_\#$, pour tous $v \in \Sigma^*_\#$, $u_1, \ldots, u_n \in \Sigma^*$ et $k \in [\![1, n]\!]$, par :

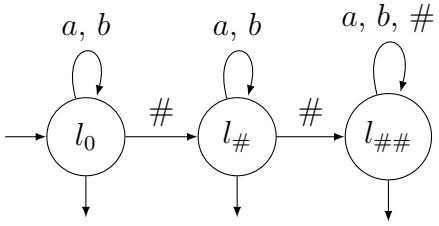
$$last^n_t(u_1\#u_2\#\ldots\#u_k) = last(u_1)\#last(u_2)\#\ldots\#last(u_k)$$
$$\text{et } last^n_t(u_1\#u_2\#\ldots\#u_n\#v) = last(u_1)\#last(u_2)\#\ldots\#last(u_n)\#v$$

Cette fonction a le même degré de séquentialité que $last^n$ car elles peuvent être réalisées par le même transducteur (à quelques légères modifications près) elle est donc bien multiséquentielle.

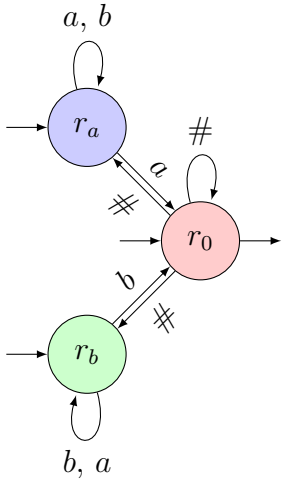
Par contre, pour $n = 2$ par exemple, l'automate droit de la bimachine canonique de $last^2_t$, représenté sur la figure ci-contre, ne permet pas d'obtenir sa décomposition. En effet, en reprenant le même raisonnement que pour $last^*$, il est clair que $last^2_{t|[r_a]}$ n'est pas à variations bornées et donc non séquentielle.

De plus, l'automate droit minimal de toutes les fonctions $last^n_t$ est le même que celui de $last^2_t$ représenté ci-contre. La seule différence entre leurs bimachines canoniques réside dans leurs automates gauches. On a alors une infinité de fonctions multiséquentielles, ayant des degrés de séquentialité différents, partageant le même automate droit minimal ce qui prouve qu'il est insuffisant de considérer seulement l'information donnée par ce dernier.

Néanmoins, il existe toujours un automate droit permettant de décomposer les fonctions multiséquentielles :



$\mathcal{L}_{last^2_t}$



$\mathcal{R}_{last^2_t}$

FIGURE 4.2 – Bimachine canonique de $last^2_t$

Proposition 4.1 *Une fonction f est multiséquentielle si et seulement s'il existe une bimachine réalisant f admettant un automate droit $\mathcal{R} = (R, I_\mathcal{R}, \{r_0\}, \Delta_\mathcal{R})$ tel que*

$$\forall r \in I_\mathcal{R}, f_{|[r]} \text{ est une fonction séquentielle} \quad (4.1)$$

Démonstration

⇒) Soient $f_1, \ldots, f_k$ des fonctions séquentielles telles que $f = \bigcup_{i=1}^k f_i$, réalisée respectivement par les transducteurs séquentiels $\mathcal{T}_1, \ldots, \mathcal{T}_k$, dont les automates sous-jacents seront notés $\mathcal{A}_j = (Q_j, \{q_{0,j}\}, F_j, \Delta_j)$ pour tout $1 \leq j \leq k$. f est donc réalisée par la réunion disjointe $\mathcal{T} = \bigsqcup_{i=1}^k \mathcal{T}_i$ dont l'automate sous-jacent sera noté $\mathcal{A} = (Q, I, F, \Delta)$.

D'après [15], il existe une bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ réalisant f où $\mathcal{R} = (R, I_\mathcal{R}, \{r_0\}, \Delta_\mathcal{R})$ est l'automate résultant de la co-déterminisation de $\mathcal{A}$.

Montrons que $\mathcal{R}$ vérifie (4.1) : Soit $r \in I_\mathcal{R}$. Il existe alors $i \in [\![1, k]\!]$ tel que $[r] \subseteq dom(f_i)$ et donc $f_{|[r]} \subseteq f_i$ d'où sa séquentialité.

En effet, pour tout mot $w = a_1 \dots a_n \in [r]$, l'unique exécution de $\mathcal{R}$ sur w est la suite $(a_{j+1} \dots a_n \cdot_{\mathcal{R}} r_0)_{j \in [\![0,n]\!]}$ où $w \cdot_{\mathcal{R}} r_0 = r$. De plus, $I = \{q_{0,j} \mid j \in [\![1,k]\!]\}$ et $r \in I_{\mathcal{R}}$ alors $I \cap r \neq \varnothing$ donc $\exists i \in [\![1,k]\!], q_{0,i} \in r$.

On a alors nécessairement $q_{0,i} \cdot_{\mathcal{A}_i} a_1 \dots a_j \in a_{j+1} \dots a_n \cdot_{\mathcal{R}} r_0$ pour tout $j \in [\![0,n]\!]$ car $\mathcal{A}$ est une réunion disjointe des $\mathcal{A}_j$ et, par définition, pour tous $a \in \Sigma$ et $s \in R$, $a \cdot_{\mathcal{R}} s = \{p \in Q \mid \exists q \in s, (p,a,q) \in \Delta\}$. Donc si $p \in Q_i \cap a \cdot_{\mathcal{R}} s$ alors $\exists q \in s$ tel que $(p,a,q) \in \Delta_i$ i.e. $p \cdot_{\mathcal{A}_i} a = q \in Q_i \cap s \subseteq s$.

En particulier, $q_{0,i} \cdot_{\mathcal{A}_i} w \in F_i$ car $r_0 = \bigsqcup_{j=1}^{k} F_j$ donc $w \in dom(f_i)$.

$\Leftarrow$) f est clairement multiséquentielle car $f = \bigcup_{s \in I_{\mathcal{R}}} f_{|[r]}$ et $I_{\mathcal{R}}$ est un ensemble fini.

■

Il est alors possible de raffiner l'automate droit minimal de toute fonction multiséquentielle afin d'obtenir un automate droit vérifiant 4.1. La figure ci-dessous, par exemple, représente deux automates droits de $last_t^2$, possédant un nombre d'état minimal, vérifiant cette condition. Les couleurs représentent les états de $\mathcal{R}_{last_t^2}$ qui ont été séparés (par exemple $[r_0] = [r_{\#a}] \cup [r_{\#b}]$).

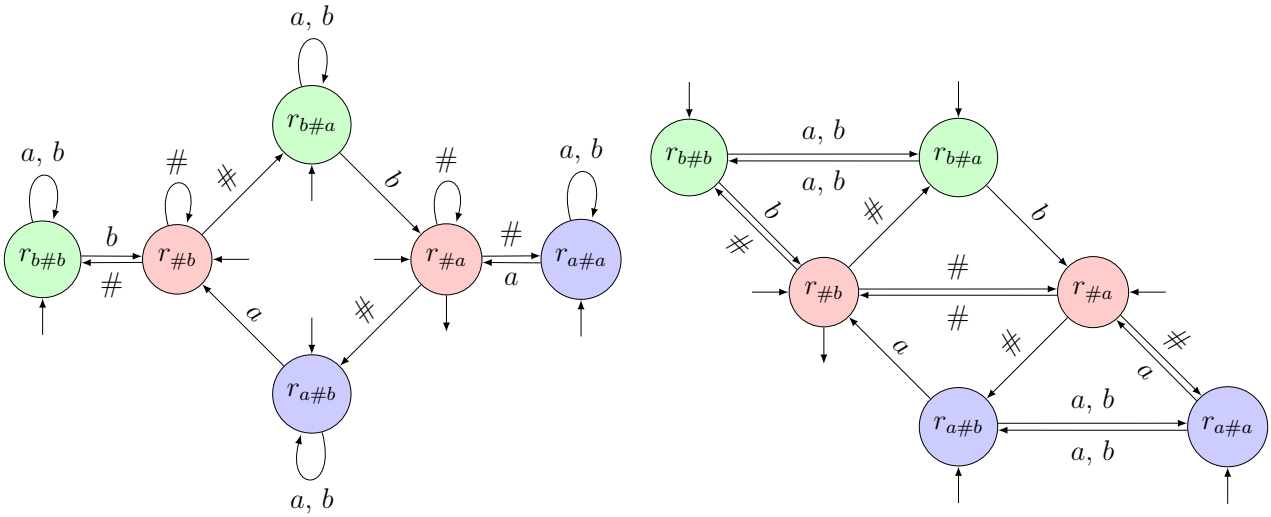


FIGURE 4.3 – Automates droits de $last_t^2$ vérifiant (4.1)

La caractérisation donnée par la proposition 4.1 est cependant insatisfaisante. Elle est trop proche de la définition des fonctions multiséquentielles et elle ne permet pas d'obtenir un automate droit vérifiant (4.1) indépendamment du transducteur réalisant la fonction donnée.

Dans le but d'obtenir une condition d'arrêt pour une éventuelle procédure de raffinement de l'automate droit minimal, nous nous sommes intéressés au degré de séquentialité des fonctions multiséquentielles. Nous avons réussi à obtenir une borne inférieure et une borne supérieure à ce degré, toutes deux exponentielles en le nombre d'états d'un transducteur réalisant la fonction. Ces résultats font l'objet de la section suivante.

4.2 Degré de séquentialité

Dans [16], les auteurs décrivent une procédure, appelée *déterminisation faible*, permettant de décomposer un transducteur réalisant une fonction multiséquentielle en réunion finie de transducteurs séquentiels. Cette construction est détaillée dans l'annexe A.

En modifiant légèrement la méthode de séparation du transducteur $\mathcal{W}(\mathcal{T})$ donné par la procédure, il est possible de réduire le nombre de transducteurs dans la réunion. Dans la construction de [16], $\mathcal{W}(\mathcal{T})$ est décomposé selon les chemins du graphe $\Psi(\mathcal{W}(\mathcal{T}))$ des composantes fortement connexes de $\mathcal{W}(\mathcal{T})$.

Si l'on contracte dans $\Psi(\mathcal{W}(\mathcal{T}))$ toutes les arêtes correspondant à une transition déterministe de $\mathcal{W}(\mathcal{T})$ (i.e. telle qu'il n'existe aucune autre transition ayant le même état de départ et la même lettre d'entrée), on obtient un nouveau graphe, que nous appellerons *graphe des composantes déterministes* de $\mathcal{W}(\mathcal{T})$ et que nous noterons $\Phi(\mathcal{W}(\mathcal{T}))$, dont les arêtes correspondent uniquement aux transitions non déterministes de $\mathcal{W}(\mathcal{T})$.

Il suffit alors de considérer la réunion sur tous les chemins p dans $\Phi(\mathcal{W}(\mathcal{T}))$, commençant dans la composante déterministe contenant l'état initial de $\mathcal{W}(\mathcal{T})$ et se terminant dans une composante contenant un état final, des transducteurs obtenus à partir de $\mathcal{W}(\mathcal{T})$ en supprimant toutes les arêtes non déterministes n'apparaissant pas dans p. D'où le résultat suivant :

Proposition 4.2 *Toute fonction multiséquentielle réalisée par un transducteur $\mathcal{T} = (\mathcal{A}, s, i, t)$, d'alphabet Σ, où $\mathcal{A} = (Q, I, F, \Delta)$, admet un degré de séquentialité au plus exponentiel en $|Q|$.*

Démonstration Les seules transitions non déterministes de $\mathcal{W}(\mathcal{T})$ sont celles crées par l'opération de séparation des transitions qui ne préservent pas le rang. Par conséquent, toutes ses composantes déterministes doivent contenir un état du type $\{(q, \epsilon)\}$ avec $q \in Q$. Donc le graphe $\Phi(\mathcal{W}(\mathcal{T}))$ a au plus $|Q|$ sommets.

On a alors au plus $2^{|Q|-1}$ choix possibles de sommets pouvant être empruntés par les chemins de $\Phi(\mathcal{W}(\mathcal{T}))$ commençant dans la composante déterministe contenant l'état initial de $\mathcal{W}(\mathcal{T})$ et se terminant dans une composante contenant un état final. Le nombre total de chemins est donc borné par $2^{|Q|-1}N^{|Q|} < (2N)^{|Q|}$ où N est le nombre maximal d'arrêtes entre les sommets de $\Phi(\mathcal{W}(\mathcal{T}))$.

Il est démontré dans [16] que, pour les fonctions multiséquentielles, la tailles des mots pouvant apparaître dans les états de $\mathcal{W}(\mathcal{T})$ est bornée (par une borne de taille polynomiale en $|Q|$). Par conséquent, si on note cette borne M, $\mathcal{W}(\mathcal{T})$ a au plus $2^{|Q|(\sum_{i=0}^{M}|\Sigma|^i)} \leq 2^{|Q|(M+1)|\Sigma|^M}$ états. On note $K = 2^{|Q|(M+1)|\Sigma|^M}$. La somme des cardinaux des composantes déterministes de $\mathcal{W}(\mathcal{T})$ est bornée par K ces composantes ne peuvent donc pas contenir plus de K états.

De plus, toutes les transitions non déterministes de $\mathcal{W}(\mathcal{T})$ doivent nécessairement mener vers un état du type $\{(q, \epsilon)\}$. C'est pourquoi, il y a au plus $|\Sigma| \cdot K$ arêtes entre deux sommets distincts de $\Phi(\mathcal{W}(\mathcal{T}))$.

Le nombre des chemins dans $\Phi(\mathcal{W}(\mathcal{T}))$, et par conséquent le degré de séquentialité, est alors borné par $(2|\Sigma| \cdot K)^{|Q|}$ d'où le résultat. ∎

Cette borne ne peut pas être améliorée. En effet, la fonction $last^n$ de l'exemple 1.6 est réalisée par le transducteur, possédant $3n$ états, représenté par la figure ci-contre et est de degré de séquentialité 2^n, soit $2^{|Q|/3}$.

Pour démontrer ce degré de séquentialité, nous aurons besoin de la propriété de Lipschitz d'ordre k, définie dans [10], caractérisant les fonctions k-séquentielles.

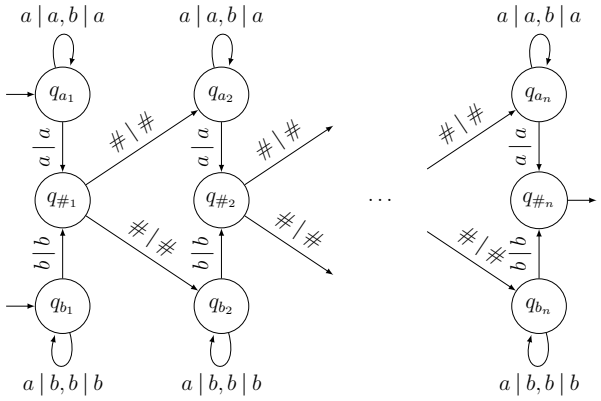


FIGURE 4.4 – Transducteur réalisant $last^n$

Définition 4.2: (Propriété de Lipschitz d'ordre k)
Une fonction f satisfait la *propriété de Lipschitz* d'ordre k s'il existe $L \in \mathbb{N}$ tel que pour tous $w_0, \dots, w_k \in dom(f)$ il existe $0 \leq i < j \leq k$ tels que $\|f(w_i), f(w_j)\| \leq L\,(\|w_i, w_j\| + 1)$

Proposition 4.3 ([10])
Une fonction satisfait la propriété de Lipschitz d'ordre k si et seulement si elle peut être réalisée par un transducteur k-séquentiel.

Il suffit alors de démontrer que $last^n$ est réalisée par un transducteur pouvant s'écrire comme une réunion de 2^n transducteurs séquentiels et qu'elle ne satisfait pas la propriété de Lipschitz d'ordre $2^n - 1$.

Considérons la suite $(u_i)_{i \in [\![0, 2^n-1]\!]}$ de mots de $\{a, b\}^n$ ordonnés lexicographiquement (où $a < b$), i.e. $u_0 = a \dots aa, u_1 = a \dots ab, \dots, u_{2^n-2} = b \dots ba, u_{2^n-1} = b \dots bb$ avec $|u_i| = n$ pour tout $i \in [\![0, 2^n - 1]\!]$. Les lettres de u_i seront notées $u_i = u_{i,1} u_{i,2} \dots u_{i,n}$.

En reprenant la notation de l'exemple 1.4 et pour tout $i \in [\![0, 2^n - 1]\!]$ on définit un transducteur séquentiel $\mathcal{T}_i$ (voir figure ci-contre) réalisant $last^n$ sur les mots dont les blocs se terminent par les lettres de u_i.

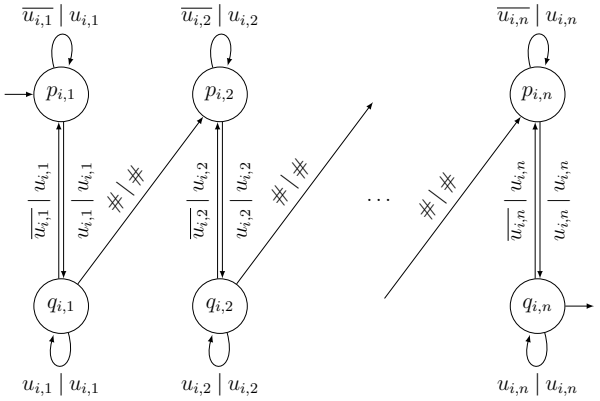


FIGURE 4.5 – Transducteur $\mathcal{T}_i$

La suite des $(u_i)_{i \in [\![0, 2^n-1]\!]}$ parcourt tous les mots de taille n. Il est alors clair que $\bigcup_{i=0}^{2^n-1} \mathcal{T}_i$ réalise $last^n$. Son degré de séquentialité est donc au plus 2^n.

Soit $L \in \mathbb{N}$ et soient $w_0, \dots, w_{2^n-1}$ les mots définis, pour tout $i \in [\![0, 2^n - 1]\!]$, par $w_i = a^{K_1} u_{i,1} \# a^{K_2} u_{i,2} \# \dots \# a^{K_n} u_{i,n}$ où $K_1, \dots, K_n \in \mathbb{N}$ vérifient $K_n > \frac{3}{2}L$ et, pour tout $l \in [\![1, n]\!]$, $K_l > \frac{1}{2}L\,(M_l + 1)$ avec $M_l = 2 + 2\,(\sum_{m=l+1}^{n} (K_m + 2))$. Ces mots permettent de contredire la propriété de Lipschitz d'ordre $2^n - 1$ pour $last^n$.

En effet, pour tous $0 \leq i < j \leq 2^n - 1$, soit l le plus petit entier de $[\![1, n]\!]$ tel que $u_{i,l} \neq u_{j,l}$. On a alors $\|u_i, u_j\| = 2 + 2\,(\sum_{m=l+1}^{n} (K_m + 2)) = M_l$ et $\|last^n(u_i), last^n(u_j)\| = 2K_l + \|u_i, u_j\| > 2K_l$. Donc par définition de K_l, $\|last^n(u_i), last^n(u_j)\| > L\,(\|w_i, w_j\| + 1)$ ce qui contredit la propriété de Lipschitz, prouvant ainsi que $last^n$ est de degré de séquentialité au moins 2^n.

La fonction $last^n$ est donc bien de degré de séquentialité 2^n, ce qui prouve que la borne exponentielle donnée par la proposition 4.2 peut être atteinte.

Chapitre 5

Minimisation de registres

Dans ce dernier chapitre, nous allons nous intéresser au problème de minimisation de registres que nous avons évoqué dans la section 3.2. Nous avons tenté de le résoudre par une correspondance entre les transducteurs à registres réalisant les fonctions rationnelles et les bimachines. Les résultats de minimisation de [13] pour les bimachines nous auraient alors permis de déterminer le nombre minimal de registres nécessaires à la réalisation de la fonction donnée. Cette correspondance s'est avérée correcte seulement pour une sous-classe de SST. Nous avons alors été amenés à modifier le modèle de bimachines afin de correspondre au cas général. Les résultats de [13] ne peuvent pas s'appliquer à ce nouveau modèle mais il permet d'avoir une autre représentation des SST.

Nous allons nous restreindre au cas des SST ayant un même registre de sortie associé à tous les états finaux, appelés SST *à registre de sortie constant*. i.e. aux SST $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ tels que $\exists X_s \in \mathcal{X}$ tel que $\forall q \in F$, $\pi_{\mathcal{X}} \circ \mu(q) = X_s$. Nous pouvons toujours nous ramener à ce cas.

Proposition 5.1 *Tout SST est équivalent à un SST à registre de sortie constant.*

Démonstration Soit $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ un SST avec $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$.

Construisons $\mathcal{T}' = (\mathcal{A}, \mathcal{X}, \mathfrak{U}', v_0', \mu')$ un SST équivalent à $\mathcal{T}$ ayant pour unique registre de sortie $X_s \in \mathcal{X}$ fixé où, $\forall \delta \in \Delta$, on notera $\mathfrak{f}'_\delta = \pi_{\mathcal{X}} \circ \mathfrak{U}'(\delta)$ et $\mathfrak{a}'_\delta = \pi_{\Sigma^*} \circ \mathfrak{U}'(\delta)$.

On définit, pour tout $q \in Q$, la fonction $\tau_q : \mathcal{X} \to \mathcal{X}$ par $\tau_q = id$ si $q \notin F$ et, si $q \in F$, $\tau_q = (\pi_{\mathcal{X}} \circ \mu(q)\ X_s)$ la transposition envoyant $\pi_{\mathcal{X}} \circ \mu(q)$ le registre de sortie associé à q sur X_s et inversement et laissant tous les autres registres fixés.

Les mises à jour de $\mathcal{T}'$ sont alors définies, pour toute transition $\delta = (p, a, q) \in \Delta$, par $\mathfrak{f}'_\delta = \tau_p \circ \mathfrak{f}_\delta \circ \tau_q$ et $\mathfrak{a}'_\delta = \mathfrak{a}_\delta \circ \tau_q$. On définit $v_0' = v_0 \circ \tau_{q_0}$ et, pour tout $q \in F$, $\mu'(q) = (X_s, \pi_{\Sigma^*} \circ \mu(q))$.

$\mathcal{T}'$ est bien un SST à registre de sortie constant et $dom([\![\mathcal{T}']\!]) = dom([\![\mathcal{T}]\!])$ car l'automate sous-jacent reste inchangé et $\mathfrak{U}(\delta)$ et $\mathfrak{U}'(\delta)$ possèdent le même domaine $\forall \delta \in \Delta$. Il reste à démontrer que $[\![\mathcal{T}']\!] = [\![\mathcal{T}]\!]$.

Soit $u = a_1 \dots a_n \in dom([\![\mathcal{T}]\!])$ et soient $(q_i, v_i)_{i \in [\![0,n]\!]}$ (resp. $(q_i, v_i')_{i \in [\![0,n]\!]}$) l'unique exécution de $\mathcal{T}$ (resp. de $\mathcal{T}'$) sur u. On notera, pour tout $i \in [\![1, n]\!]$, $\delta_i = (q_{i-1}, a_i, q_i) \in \Delta$. On notera aussi $X_n = \pi_{\mathcal{X}} \circ \mu(q_n)$ (resp. $X'_n = X_s$) et, pour tout $i \in [\![1, n]\!]$, $X_{i-1} = \mathfrak{f}_{\delta_i}(X_i)$ (resp. $X'_{i-1} = \mathfrak{f}'_{\delta_i}(X'_i)$)

On remarque que, pour tout $i \in [\![0, n]\!]$, $X'_{n-i} = \tau_{q_{n-i}}(X_{n-i})$. En effet, par récurrence sur i,

$X'_n = X_s = \tau_{q_n}(\pi_{\mathcal{X}} \circ \mu(q_n)) = \tau_{q_n}(X_n)$ et si $X'_{n-i+1} = \tau_{q_{n-i+1}}(X_{n-i+1})$ alors

$$\begin{aligned} X'_{n-i} &= \mathfrak{f}'_{\delta_{n-i+1}}(X'_{n-i+1}) = \tau_{q_{n-i}} \circ \mathfrak{f}_{\delta_{n-i+1}} \circ \tau_{q_{n-i+1}}(X'_{n-i+1}) \\ &= \tau_{q_{n-i}} \circ \mathfrak{f}_{\delta_{n-i+1}} \circ \tau_{q_{n-i+1}}(\tau_{q_{n-i+1}}(X_{n-i+1})) = \tau_{q_{n-i}} \circ \mathfrak{f}_{\delta_{n-i+1}}(X_{n-i+1}) \\ X'_{n-i} &= \tau_{q_{n-i}}(X_{n-i}) \end{aligned}$$

Par définition, $[\![\mathcal{T}]\!](u) = v_n(X_n)\,\pi_{\Sigma^*} \circ \mu\,(q_n)$ et $[\![\mathcal{T}']\!](u) = v'_n(X'_n)\,\pi_{\Sigma^*} \circ \mu\,(q_n)$. Il suffit alors de démontrer que $v'_n(X'_n) = v_n(X_n)$.

D'une part $v_n(X_n) = v_{n-1}(X_{n-1})\mathfrak{a}_{\delta_n}(X_n) = \cdots = v_0(X_s)\prod_{i=1}^{n}\mathfrak{a}_{\delta_i}(X_i)$, d'autre part

$$\begin{aligned} v'_n(X'_n) &= v'_{n-1}(X'_{n-1})\mathfrak{a}'_{\delta_n}(X'_n) = \cdots = v'_0(X'_s)\prod_{i=1}^{n}\mathfrak{a}'_{\delta_i}(X'_i) \\ &= v_0 \circ \tau_{q_0}(\tau_{q_0}(X_s))\prod_{i=1}^{n}\mathfrak{a}_{\delta_i} \circ \tau_{q_i}(\tau_{q_i}(X_i)) \\ v'_n(X'_n) &= v_0(X_s)\prod_{i=1}^{n}\mathfrak{a}_{\delta_i}(X_i) = v_n(X_n) \end{aligned}$$

■

5.1 SST à flows indépendants

Définition 5.1: On dit qu'un SST $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ est *à flows indépendants* si le flow de toute transition de $\mathcal{T}$ est constant et ne dépend pas de ses états,

i.e. pour toute lettre a, $\exists g_a \in \mathcal{F}(\mathcal{X}, \mathcal{X})$ telle que $\forall \delta = (p, a, q) \in \Delta$, $\mathfrak{f}_\delta = g_a$.

Proposition 5.2 *L'ensemble des SST à flows indépendants réalisant une fonction rationnelle est en bijection avec l'ensemble des bimachines réalisant cette fonction. Cette bijection associe à tout SST une bimachine dont l'automate gauche est l'automate sous-jacent du SST et dont le nombre d'états de l'automate droit correspond au nombre de registres.*

5.1.1 Construction des bimachines correspondantes

Nous allons décrire ici une procédure permettant de transformer tout SST à flows indépendants en une bimachine et inversement, prouvant ainsi la proposition précédente.

SST vers bimachine

Soit $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$, où $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$, un SST à flows indépendants et à registre de sortie constant $X_s \in \mathcal{X}$. Construisons une bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$ équivalente à $\mathcal{T}$, où $\mathcal{L} = (L, \{l_0\}, F_{\mathcal{L}}, \Delta_{\mathcal{L}})$ et $\mathcal{R} = (R, I_{\mathcal{R}}, \{r_0\}, \Delta_{\mathcal{R}})$:

— L'automate gauche de $\mathcal{B}$ est l'automate sous-jacent de $\mathcal{T}$, i.e. $\mathcal{L} = \mathcal{A}$.

— L'automate droit $\mathcal{R}$ est défini en fonction des registres par $R = \mathcal{X}$, $I_{\mathcal{R}} = dom(v_0)$, $r_0 = X_s$ et $\Delta_{\mathcal{R}} = \{(\mathfrak{f}_\delta(r), a, r)$ où $\delta = (p, a, q) \in \Delta\}$ (Par hypothèse, $\mathfrak{f}_\delta$ ne dépend que de a, donc $a \cdot_{\mathcal{R}} r$ est bien défini).

— On définit $\omega(l, \sigma, r) = \mathfrak{a}_\delta(r)$ où $\delta = (l, \sigma, l \cdot_{\mathcal{L}} \sigma) \in \Delta$, pour tous $l \in L$, $r \in R$ et $\sigma \in \Sigma$, tels que $l \cdot_{\mathcal{L}} \sigma$ et $\sigma \cdot_{\mathcal{R}} r$ sont définis.

— Pour tout $r \in I_{\mathcal{R}}$, on définit $\lambda(r) = v_0(r)$.
— Pour tout $l \in F_{\mathcal{L}}$, on définit $\rho(l) = \pi_{\Sigma^*} \circ \mu(l)$.

On a alors $[\![\mathcal{B}]\!] = [\![\mathcal{T}]\!]$.

Démonstration Soit $u = a_1 \dots a_n \in dom([\![\mathcal{B}]\!]) \cap dom([\![\mathcal{T}]\!])$. Montrons que $[\![\mathcal{T}]\!](u) = [\![\mathcal{B}]\!](u)$.

Considérons l'exécution acceptante $(q_i, v_i)_{i\in[\![0,n]\!]}$ de $\mathcal{T}$ sur u. Si on note $\forall i \in [\![1,n]\!]$, $\delta_i = (q_{i-1}, a_i, q_i) \in \Delta$, $X_{i-1} = \mathfrak{f}_{\delta_i}(X_i)$ avec $X_n = X_s$ et $w_i = \mathfrak{a}_{\delta_i}(X_i)$.

On a pour tout $i \in [\![1,n]\!]$, par définition, $\begin{cases} q_i = q_{i-1} \cdot_{\mathcal{L}} a_i \\ X_{i-1} = a_i \cdot_{\mathcal{R}} X_i \\ \omega(q_{i-1}, a_i, X_i) = w_i \end{cases}$

Donc pour tout $i \in [\![1,n]\!]$, $\begin{cases} q_i = q_0 \cdot a_1 \dots a_i \\ X_{i-1} = a_i \dots a_n \cdot X_s \end{cases}$ et $\omega(q_0 \cdot a_1 \dots a_{i-1},\, a_i,\, a_{i+1} \dots a_n \cdot X_s) = w_i$.

On a alors

$$\begin{aligned} v_n(X_s) &= v_{n-1}(X_{n-1})w_n = v_{n-2}(X_{n-2})w_{n-1}w_n = \dots = v_0(X_s) \prod_{i=1}^{n} w_i \\ &= v_0(u \cdot X_s) \prod_{i=1}^{n} \omega(q_0 \cdot a_1 \dots a_{i-1},\, a_i,\, a_{i+1} \dots a_n \cdot X_s) \\ v_n(X_s) &= \lambda(u \cdot X_s)\, \omega(q_0, u, X_s) \end{aligned}$$

Donc

$$[\![\mathcal{T}]\!](u) = v_n(X_s)\, \pi_{\Sigma^*} \circ \mu(q_n) = \lambda(u \cdot X_s)\, \omega(q_0, u, X_s)\, \rho(q_0 \cdot u) = [\![\mathcal{B}]\!](u)$$

Il reste à démontrer que $dom([\![\mathcal{B}]\!]) = dom([\![\mathcal{T}]\!])$. Si $u \in dom([\![\mathcal{B}]\!])$ alors $u \in dom([\![\mathcal{T}]\!])$.

En effet, nous pouvons considérer l'exécution acceptante $(q_i, v_i)_{i\in[\![0,n]\!]}$ de $\mathcal{T}$ sur u définie, pour tout $i \in [\![1,n]\!]$, par $q_i = l_0 \cdot a_1 \dots a_i$ et $v_i(r) = \lambda(a_1 \dots a_i \cdot r) \prod_{j=1}^{i} \omega(l_0 \cdot a_1 \dots a_{j-1}, a_j, a_{j+1} \dots a_i \cdot r)$, $\forall r \in dom(v_i)$ avec

$$dom(v_i) = \left((a_1 \dots a_i)^{-1} I_{\mathcal{R}}\right) \cap \left(\bigcap_{j=1}^{i} (a_{j+1} \dots a_i)^{-1} dom(\omega(l_0 \cdot a_1 \dots a_{j-1}, a_j, -)) \right)$$

où $(a_1 \dots a_i)^{-1} I_{\mathcal{R}} = \{r \in R \,|\, a_1 \dots a_i \cdot r \in I_{\mathcal{R}}\}$ et $(a_{j+1} \dots a_i)^{-1} dom(\omega(l_0 \cdot a_1 \dots a_{j-1}, a_j, -)) = \{r \in R \,|\, (l_0 \cdot a_1 \dots a_{j-1}, a_j, a_{j+1} \dots a_i \cdot r) \in dom(\omega)\}$.

Cette suite définit bien une exécution de $\mathcal{T}$ sur u car, $\forall i \in [\![1,n]\!]$

— $\delta_i = (q_{i-1}, a_i, q_i) = (q_{i-1}, a_i, q_{i-1} \cdot_{\mathcal{L}} a_i) \in \Delta$.
— $dom(v_i) = a_i^{-1} \cdot_{\mathcal{R}} dom(v_{i-1}) = \mathfrak{f}_{\delta_i}^{-1}(dom(v_{i-1}))$
— $\forall r \in dom(v_i)$, $v_i(r) = v_{i-1}(a_i \cdot_{\mathcal{R}} r)\omega(q_{i-1}, a_i, r) = v_{i-1}(\mathfrak{f}_{\delta_i}(r))\, \mathfrak{a}_{\delta_i}(r)$

Cette exécution est acceptante car $q_n = l_0 \cdot u \in F_{\mathcal{L}} = F$ et $\pi_{\mathcal{X}} \circ \mu(q_n) = X_s = r_0 \in dom(v_n)$ car $u \in dom([\![\mathcal{B}]\!])$.

Réciproquement, si $u \in dom([\![\mathcal{T}]\!])$, soit $(q_i, v_i)_{i\in[\![0,n]\!]}$ l'unique exécution de $\mathcal{T}$ sur u. Par définition, pour tout $i \in [\![1,n]\!]$, $\delta_i = (q_{i-1}, a_i, q_i) \in \Delta = \Delta_{\mathcal{L}}$ donc $q_i = q_{i-1} \cdot_{\mathcal{L}} a_i$ est défini et, en particulier, tous les $q_i = l_0 \cdot a_1 \dots a_i$ sont définis et $q_n = l_0 \cdot u \in F = F_{\mathcal{L}}$.

De plus, pour tout $r \in dom(v_i)$, $v_i(r) = v_{i-1}(\mathfrak{f}_{\delta_i}(r))\mathfrak{a}_{\delta_i}(r) = v_{i-1}(a_i \cdot_{\mathcal{R}} r)\, \omega(q_{i-1}, a_i, r)$. Donc $a_i \cdot_{\mathcal{R}} r$ et $\omega(q_{i-1}, a_i, r)$ sont définis. En particulier, $a_{i+1} \dots a_n \cdot r_0 \in dom(v_i)$ $\forall i \in [\![0,n]\!]$, car $\pi_{\mathcal{X}} \circ \mu(q_n) = r_0 \in dom(v_n)$ et $\mathfrak{f}_{\delta_i}(dom(v_i)) = a_i \cdot_{\mathcal{R}} dom(v_i) \subseteq dom(v_{i-1})$. Donc $a_n \cdot_{\mathcal{R}} r_0 \in dom(v_{n-1})$ et $a_{n-1} \cdot_{\mathcal{R}} (a_n \cdot_{\mathcal{R}} r_0) \in dom(v_{n-2})$ et … et $a_{i+1} \dots a_n \cdot r_0 \in dom(v_i)$.

i.e. tous les $a_{i+1} \dots a_n \cdot r_0$ et $\omega(l_0 \cdot a_1 \dots a_{i-1}, a_i, a_{i+1} \dots a_n \cdot r_0)$ sont définis et $u \cdot r_0 \in dom(v_0) = I_{\mathcal{R}}$. Donc $u \in dom([\![B]\!])$ et les domaines sont donc bien égaux. ∎

Bimachine vers SST

Considérons maintenant une bimachine $\mathcal{B} = (\mathcal{L}, \mathcal{R}, \omega, \lambda, \rho)$, où $\mathcal{L} = (L, \{l_0\}, F_\mathcal{L}, \Delta_\mathcal{L})$ et $\mathcal{R} = (R, I_\mathcal{R}, \{r_0\}, \Delta_\mathcal{R})$ et construisons un SST $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ équivalent à $\mathcal{B}$, avec $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$. La construction est symétrique à la précédente :

— L'automate sous-jacent de $\mathcal{T}$ est l'automate gauche de $\mathcal{B}$, i.e. $\mathcal{A} = \mathcal{L}$.

— Les registres et leurs mises à jour sont construits à partir de $\mathcal{R}$: $\mathcal{X} = R$ et on définit $\mathfrak{U}(\delta)(r) = (a \cdot_\mathcal{R} r, \omega(l, a, r))$, pour tous $\delta = (l, a, l \cdot_\mathcal{L} a) \in \Delta$ et $r \in dom(a \cdot_\mathcal{R} -) \cap dom(\omega(l, a, -))$.

— Pour tout $r \in I_\mathcal{R}$ on définit $v_0(r) = \lambda(r)$.

— Pour tout $l \in F_\mathcal{L} = F$ on définit $\mu(l) = (r_0, \rho(l))$ et on notera le registre de sortie $X_s = r_0$.

On a bien $[\![\mathcal{T}]\!] = [\![\mathcal{B}]\!]$.

Démonstration La preuve de l'égalité des domaines et la construction de l'exécution de $\mathcal{T}$ à partir de celle de $\mathcal{B}$ reste la même que dans la démonstration précédente.

En reprenant les mêmes notations, on trouve que :

$$\begin{aligned}
[\![\mathcal{T}]\!](u) &= v_n(\pi_\mathcal{X} \circ \mu(q_n))\, \pi_{\Sigma^*} \circ \mu(q_n) = v_n(r_0)\rho(q_n) \\
&= \lambda(u \cdot r_0) \prod_{i=1}^{n} \omega(l_0 \cdot a_1 \dots a_{i-1}, a_i, a_{i+1} \dots a_n \cdot r_0)\ \rho(l_0 \cdot u) \\
[\![\mathcal{T}]\!](u) &= [\![\mathcal{B}]\!](u)
\end{aligned}$$

■

Cette correspondance nous permet alors d'obtenir le résultat suivant :

Théorème 5.3 *Soit f une fonction rationnelle réalisée par un SST à flows indépendants et à registre de sortie constant.*

Le nombre minimal de registres nécessaires à la réalisation de f est égale au nombre d'états de l'automate droit minimal $\mathcal{R}_f$ qui lui est associé.

Plus concrètement, il est possible de transformer le SST donné en bimachine, puis, d'appliquer la procédure de minimisation des bimachines décrite dans [13], et enfin de construire le SST correspondant à cette bimachine minimale.

Il est à noter qu'il est possible de minimiser l'automate gauche de la bimachine afin d'obtenir un SST possédant un automate sous-jacent minimal mais ceci peut engendrer une croissance du nombre de registres. En effet, la taille de l'automate droit d'une bimachine est généralement inversement proportionnelle à celle de son automate gauche et vice versa. Cela suggère une relation inversement proportionnelle entre le nombre de registres d'un SST et la taille de son automate sous-jacent.

5.2 Bimachines asynchrones

Nous pouvons ignorer la contrainte d'indépendance des flows en modifiant le modèle de bimachines considéré. Nous appellerons ce nouveau modèle *bimachine asynchrone* car il peut être vu comme une bimachine dont l'automate droit est exécuté après la fin de l'exécution de l'automate gauche. Il est constitué d'un transducteur séquentiel $\mathcal{S}$ étiquetant les entrées par ses états et d'un transducteur co-séquentiel $\mathcal{C}$, défini sur les mots étiquetés, lisant l'entrée de droite à gauche et produisant la sortie.

Définition 5.2: Une *bimachine asynchrone*, d'alphabet Σ, est un 5-uplet $\mathcal{B} = (\mathcal{S}, \mathcal{C}, s, i, t)$ où :

- $\mathcal{S} = (Q_\mathcal{S}, \{q_0\}, F_\mathcal{S}, \Delta_\mathcal{S})$ est un automate fini déterministe sur l'alphabet Σ, i.e. $\Delta_\mathcal{S} \subseteq Q_\mathcal{S} \times \Sigma \times Q_\mathcal{S}$
- $\mathcal{C} = (Q_\mathcal{C}, I_\mathcal{C}, \{q_f\}, \Delta_\mathcal{C})$ est un automate fini co-déterministe sur l'alphabet $\Sigma \times Q_\mathcal{S}$, i.e. $\Delta_\mathcal{C} \subseteq Q_\mathcal{C} \times (\Sigma \times Q_\mathcal{S}) \times Q_\mathcal{C}$
- $s : \Delta_\mathcal{C} \to \Sigma^*$ est la fonction de sortie.
- $i : I_\mathcal{C} \to \Sigma^*$ est la fonction de sortie initiale.
- $t : F_\mathcal{S} \to \Sigma^*$ est la fonction de sortie terminale.

Une *exécution* de $\mathcal{B}$ sur un mot $u = a_1 \dots a_n \in \Sigma^*$ est une suite finie d'éléments de $Q_\mathcal{S} \times Q_\mathcal{C}$, $(q_i, p_i)_{i \in [\![0,n]\!]}$ telle que pour tout $i \in [\![1,n]\!]$, $(q_{i-1}, a_i, q_i) \in \Delta_\mathcal{S}$ et $\delta_i = (p_{i-1}, (a_i, q_{i-1}), p_i) \in \Delta_\mathcal{C}$.

L'exécution est *acceptante* si q_0 est l'état initial de $\mathcal{S}$ (resp. $p_n = q_f$ est l'état final de $\mathcal{C}$) et $q_n \in F_\mathcal{S}$ (resp. $p_0 \in I_\mathcal{C}$).

$\mathcal{B}$ réalise une fonction $[\![\mathcal{B}]\!] : \Sigma^* \to \Sigma^*$ définie, sur les mots $u \in \Sigma^*$ tels qu'il existe une exécution acceptante $(q_i, p_i)_{i \in [\![0,n]\!]}$ de $\mathcal{B}$ sur u, par

$$[\![\mathcal{B}]\!](u) = i(p_0) \prod_{i=1}^{n} s(\delta_i)\, t(q_n)$$

5.2.1 Construction des SST correspondants

Nous pouvons maintenant démontrer, par une construction similaire à la précédente, la correspondance entre les bimachines asynchrones et les SST réalisant les fonctions rationnelles.

SST vers bimachine asynchrone

Soit $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$, où $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$, un SST à registre de sortie constant $X_s \in \mathcal{X}$. Construisons une bimachine asynchrone $\mathcal{B} = (\mathcal{S}, \mathcal{C}, s, i, t)$ équivalente à $\mathcal{T}$, où $\mathcal{S} = (Q_\mathcal{S}, \{q_0\}, F_\mathcal{S}, \Delta_\mathcal{S})$ et $\mathcal{C} = (Q_\mathcal{C}, I_\mathcal{C}, \{q_f\}, \Delta_\mathcal{C})$:

- L'automate déterministe de $\mathcal{B}$ est l'automate sous-jacent de $\mathcal{T}$. i.e. $\mathcal{S} = \mathcal{A}$.
- L'automate co-déterministe $\mathcal{C}$ est défini en fonction des registres par $Q_\mathcal{C} = \mathcal{X}$, $I_\mathcal{C} = dom(v_0)$, $q_f = X_s$ et $\Delta_\mathcal{C} = \{(Y, (a,p), X) \mid \exists\, \delta = (p, a, q) \in \Delta \text{ telle que } \mathfrak{f}_\delta(X) = Y\}$ ($\mathcal{A}$ étant déterministe, la transition δ est unique. $\mathcal{C}$ est donc bien co-déterministe car $\mathfrak{f}_\delta$ est une fonction). Les transitions étant étiquetées par les états de $\mathcal{A}$, l'hypothèse d'indépendance des flows n'est plus nécessaire.
- Pour toute transition $\delta_\mathcal{C} = (Y, (a,p), X) \in \Delta_\mathcal{C}$ associée à une transition $\delta = (p, a, q) \in \Delta$, on définit $s(\delta_\mathcal{C}) = \mathfrak{a}_\delta(X)$.
- Pour tout $X \in I_\mathcal{C}$, on définit $i(X) = v_0(X)$.
- Pour tout $q \in F_\mathcal{S}$, on définit $t(q) = \pi_{\Sigma^*} \circ \mu(q)$.

On a alors $[\![\mathcal{B}]\!] = [\![\mathcal{T}]\!]$.

Démonstration Soit $u = a_1 \dots a_n \in dom([\![\mathcal{T}]\!])$. Reprenons les mêmes notations que dans la démonstration de la construction précédente. i.e. $(q_i, v_i)_{i \in [\![0,n]\!]}$ désigne l'exécution de $\mathcal{T}$ sur u, $\forall i \in [\![1,n]\!]$, $\delta_i = (q_{i-1}, a_i, q_i) \in \Delta$, $X_{i-1} = \mathfrak{f}_{\delta_i}(X_i)$ avec $X_n = X_s$ et $w_i = \mathfrak{a}_{\delta_i}(X_i)$.

La suite $(q_i, X_i)_{i \in [\![0,n]\!]}$ est alors l'exécution acceptante de $\mathcal{B}$ sur u (donc, en particulier, $u \in dom([\![\mathcal{B}]\!])$). En effet, $\mathcal{S} = \mathcal{A}$ donc $(q_i)_{i \in [\![0,n]\!]}$ est une exécution acceptante de $\mathcal{S}$ sur u et, par définition, $X_n = X_s = q_f$, $X_0 \in dom(v_0) = I_\mathcal{C}$ et $\delta'_i = (X_{i-1}, (a_i, q_{i-1}), X_i) \in \Delta_\mathcal{C}$ pour tout $i \in [\![1,n]\!]$.

On a alors $v_n(X_s) = v_0(X_s)\ \prod_{i=1}^{n} w_i = i(X_0)\ \prod_{i=1}^{n} s(\delta'_i)$ et donc

$$[\![\mathcal{T}]\!](u) = v_n(X_s)\,\pi_{\Sigma^*} \circ \mu\,(q_n) = i(X_0)\ \prod_{i=1}^{n} s(\delta'_i)\ t(q_n) = [\![\mathcal{B}]\!](u)$$

La preuve de l'inclusion de $dom([\![\mathcal{B}]\!])$ dans $dom([\![\mathcal{T}]\!])$ est analogue au cas des bimachines. Elle vient du fait que si $(q_i, X_i)_{i\in[\![0,n]\!]}$ est l'exécution acceptante de $\mathcal{B}$ sur un mot $u = a_1 \dots a_n \in dom([\![\mathcal{B}]\!])$, il est possible de construire une exécution acceptante $(q_i, v_i)_{i\in[\![0,n]\!]}$ de $\mathcal{T}$ sur u. Si on note, pour tout $i \in [\![1, n]\!]$, $\alpha_i = (a_i, q_{i-1})$ et $\delta'_i = (X_{i-1}, \alpha_i, X_i) \in \Delta_{\mathcal{C}}$, on peut définir v_i, pour tout registre $X \in dom(v_i)$, par $v_i(X) = i(\alpha_1 \dots \alpha_i \cdot_{\mathcal{C}} X) \prod_{j=1}^{i} s(\alpha_j \dots \alpha_i \cdot_{\mathcal{C}} X \xrightarrow{\alpha_j} \alpha_{j+1} \dots \alpha_i \cdot_{\mathcal{C}} X)$ avec

$$dom(v_i) = \left((\alpha_1 \dots \alpha_i)^{-1} I_{\mathcal{C}}\right) \cap \left(\bigcap_{j=1}^{i} (\alpha_{j+1} \dots \alpha_i)^{-1} \Delta_{\mathcal{C},\alpha_j}\right)$$

où $(\alpha_1 \dots \alpha_i)^{-1} I_{\mathcal{C}} = \{X \in Q_{\mathcal{C}} \,|\, \alpha_1 \dots \alpha_i \cdot X \in I_{\mathcal{C}}\}$
et $(\alpha_{j+1} \dots \alpha_i)^{-1} \Delta_{\mathcal{C},\alpha_j} = \{X \in Q_{\mathcal{C}} \,|\, \alpha_j \cdot_{\mathcal{C}} (\alpha_{j+1} \dots \alpha_i \cdot_{\mathcal{C}} X) \text{ est défini}\}$.

Cette suite définit bien une exécution de $\mathcal{T}$ sur u car, $\forall i \in [\![1, n]\!]$

— $\delta_i = (q_{i-1}, a_i, q_i) = (q_{i-1}, a_i, q_{i-1} \cdot_{\mathcal{L}} a_i) \in \Delta$.

— $dom(v_i) = \alpha_i^{-1} \cdot_{\mathcal{C}} dom(v_{i-1}) = \mathfrak{f}_{\delta_i}^{-1}(dom(v_{i-1}))$

— $\forall X \in dom(v_i),\ v_i(X) = v_{i-1}(\alpha_i \cdot_{\mathcal{C}} X) s(\alpha_i \cdot_{\mathcal{C}} X \xrightarrow{\alpha_i} X) = v_{i-1}(\mathfrak{f}_{\delta_i}(X))\,\mathfrak{a}_{\delta_i}(X)$

Cette exécution est acceptante car $q_n \in F_{\mathcal{S}} = F$ et $\pi_{\mathcal{X}} \circ \mu\,(q_n) = X_s = q_f = X_n \in dom(v_n)$ car $u \in dom([\![\mathcal{B}]\!])$. ■

Bimachine asynchrone vers SST

Comme précédemment, nous avons une construction symétrique permettant de passer d'une bimachine asynchrone à un SST. Soit $\mathcal{B} = (\mathcal{S}, \mathcal{C}, s, i, t)$ une bimachine asynchrone, avec $\mathcal{S} = (Q_{\mathcal{S}}, \{q_0\}, F_{\mathcal{S}}, \Delta_{\mathcal{S}})$ et $\mathcal{C} = (Q_{\mathcal{C}}, I_{\mathcal{C}}, \{q_f\}, \Delta_{\mathcal{C}})$. Construisons un SST $\mathcal{T} = (\mathcal{A}, \mathcal{X}, \mathfrak{U}, v_0, \mu)$ équivalent à $\mathcal{B}$, où $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$:

— L'automate sous-jacent de $\mathcal{T}$ est l'automate déterministe de $\mathcal{B}$. i.e. $\mathcal{A} = \mathcal{S}$.

— Les registres et leurs mises à jour sont construits à partir de $\mathcal{C}$: $\mathcal{X} = Q_{\mathcal{C}}$ et on définit, pour toute transition $\delta = (p, a, q) \in \Delta$, $\mathfrak{U}(\delta)(X) = (Y, s(\delta'))$, sur les $X \in Q_{\mathcal{C}}$ tels que $\exists \delta' = (Y, (a, p), X) \in \Delta_{\mathcal{C}}$.

— Pour tout $X \in I_{\mathcal{C}}$ on définit $v_0(X) = i(X)$.

— Pour tout $q \in F_{\mathcal{S}} = F$ on définit $\mu(q) = (q_f, t(q))$ et on notera le registre de sortie $X_s = q_f$.

On a bien $[\![\mathcal{T}]\!] = [\![\mathcal{B}]\!]$.

Démonstration Comme pour les bimachines, la preuve de l'égalité des domaines et la construction de l'exécution de $\mathcal{T}$ à partir de celle de $\mathcal{B}$ reste la même que dans la démonstration précédente et en reprenant les mêmes notations, on trouve que :

$$\begin{aligned}[\![\mathcal{T}]\!](u) &= v_n(\pi_{\mathcal{X}} \circ \mu\,(q_n))\,\pi_{\Sigma^*} \circ \mu\,(q_n) = v_n(X_n)\,t(q_n) \\ &= i(\alpha_1 \dots \alpha_n \cdot_{\mathcal{C}} X_n) \prod_{i=1}^{n} s(\alpha_i \dots \alpha_n \cdot_{\mathcal{C}} X_n \xrightarrow{\alpha_i} \alpha_{i+1} \dots \alpha_n \cdot_{\mathcal{C}} X_n)\ t(q_n) \\ [\![\mathcal{T}]\!](u) &= [\![\mathcal{B}]\!](u)\end{aligned}$$

■

Conclusion

Dans ce travail, nous avons abordé plusieurs problèmes concernant les fonctions rationnelles par une approche basée sur les bimachines. Nous avons obtenu les résultats suivants :

— Pour la sous-classe des fonctions multiséquentielles, nous avons démontré que l'automate droit était insuffisant pour décider de l'appartenance d'une fonction donnée à cette sous-classe.

— Nous avons obtenu une borne exponentielle sur le degré de séquentialité.

— Nous nous sommes intéressés au problème de la minimisation de registres des SST et nous avons obtenu un résultat positif grâce à la correspondance entre les bimachines et les SST à flows indépendants.

— Nous avons introduit le modèle de bimachines asynchrones, correspondant aux SST réalisant les fonctions rationnelles.

Deux problèmes intéressants restent encore ouverts :

— Le premier concerne la caractérisation algébrique des fonctions multiséquentielles. Nous avons montré, dans le quatrième chapitre, qu'il existait toujours un automate droit permettant de décomposer le domaine des fonctions multiséquentielles en parties pouvant être traitées séquentiellement. Il serait intéressant de trouver une procédure permettant de raffiner l'automate droit minimal afin d'obtenir un automate droit vérifiant cette propriété.

— Le second problème est celui de la minimisation des bimachines asynchrones. Il faudrait réussir à appliquer les méthodes de [7] ou [13] à ce modèle alternatif de bimachines. Cela permettrait de résoudre le problème de la minimisation de registres dans le cas général des SST réalisant les fonctions rationnelles.

Annexe A

Procédure de déterminisation faible

Soit $\mathcal{T} = (\mathcal{A}, s, i, t)$, où $\mathcal{A} = (Q, I, F, \Delta)$, un transducteur fonctionnel.

A.1 Déterminisation

Décrivons, en premier lieu, la procédure de déterminisation, introduite dans [3], permettant d'obtenir un transducteur séquentiel $\mathcal{D}(\mathcal{T})$ équivalent à $\mathcal{T}$, dans le cas où $[\![\mathcal{T}]\!]$ est une fonction séquentielle.

Cette construction généralise la construction par sous-ensembles d'un automate déterministe. Les états de $\mathcal{D}(\mathcal{T})$ sont des ensembles de couples $(q, w) \in Q \times \Sigma^*$. En lisant un mot u, $\mathcal{D}(\mathcal{T})$ produit le plus long préfixe commun des sorties des différentes exécutions de $\mathcal{T}$ sur u, notons le w, et atteint un état contenant, pour chaque exécution, un couple dont la première composante est l'état de $\mathcal{T}$ qu'elle permet d'atteindre et dont la deuxième est le mot restant après avoir retiré w à la sortie de cette exécution.

On notera $\mathcal{D}^\infty(\mathcal{T}) = (\mathcal{A}', s', i', t')$, où $\mathcal{A}' = (Q', \{q'_0\}, F', \Delta')$ est un automate déterministe (à ensemble d'états infini). On prend alors :

— $Q' = \mathcal{P}_f(Q \times \Sigma^*)$ l'ensemble des parties finies de $Q \times \Sigma^*$.

— $q'_0 = \{(q, w_0^{-1} i(q)) \mid q \in I\}$ où $w_0 = \bigwedge_{q \in I} i(q)$.

— $F' = \{q' \in Q' \mid \pi_Q(q') \cap F \neq \varnothing\}$.

— Pour tous $p' \in Q'$ et $\sigma \in \Sigma$, on définit :

$$R_{p',\sigma} = \left\{ (q, uv) \in Q \times \Sigma^* \;\middle|\; \exists (p, u) \in p' \text{ tel que } p \xrightarrow{\sigma|v}_{\mathcal{T}} q \right\}$$

$$w_{p',\sigma} = \bigwedge \pi_{\Sigma^*}(R_{p',\sigma})$$

Les transitions de $\mathcal{D}^\infty(\mathcal{T})$ sont alors définies par :

$$p' \cdot \sigma = \left\{ (q, w_{p',\sigma}^{-1} w) \in Q \times \Sigma^* \;\middle|\; (q, w) \in R_{p',\sigma} \right\}$$

Pour les sorties :

— Pour toute transition $\delta = (p', \sigma, p' \cdot \sigma) \in \Delta$, on définit $s'(\delta) = w_{p',\sigma}$.

— La sortie initiale est définie par $i'(q'_0) = w_0 = \bigwedge_{q \in I} i(q)$

— La sortie terminale est définie, pour tout $q' \in F'$, par $\mathfrak{t}'(q') = w\mathfrak{t}(q)$ où $(q, w) \in q' \cap (F \times \Sigma^*)$ ($\mathcal{T}$ étant fonctionnel, cette valeur ne dépend pas du choix de (q, w)).

Il a alors été démontré dans [3] que $\mathcal{T}$ et $\mathcal{D}^\infty(\mathcal{T})$ sont équivalents. De plus, si on note $\mathcal{D}(\mathcal{T})$ le transducteur obtenu à partir de $\mathcal{D}^\infty(\mathcal{T})$ en ne gardant que les états accessibles et co-accessibles et si $[\![\mathcal{T}]\!]$ est une fonction séquentielle, alors $\mathcal{D}(\mathcal{T})$ est un transducteur fini, séquentiel et équivalent à $\mathcal{T}$.

A.2 Déterminisation faible

Dans le cas des fonctions non séquentielles, la procédure décrite ci-dessus produit un transducteur infini. Elle peut cependant être adaptée au cas des fonctions multiséquentielle afin d'obtenir une décomposition de $\mathcal{T}$ en union disjointe de transducteurs séquentiels. Cette nouvelle procédure a été décrite, et appelée *déterminisation faible*, dans [16]. On suppose que $\mathcal{T}$ possède un unique état initial.

Deux états p et q de $\mathcal{T}$ sont dits *fortement connectés*, on notera $p \sim q$, s'il existe une exécution de $\mathcal{T}$ de p à q et inversement. Les classes d'équivalence pour la relation $\sim$ sont appelées les *composantes fortement connexes* de $\mathcal{T}$. Une transition reliant deux états dans deux composantes fortement connexes différentes est dite *transitoire*.

Dans la construction précédente, pour tout état $q' \in \mathcal{P}_f(Q \times \Sigma^*)$, on définit son *rang*, noté $rg(q')$, comme l'ensemble des composantes fortement connexes de $\mathcal{T}$ accessibles à partir des états de $\pi_Q(q')$.

La déterminisation faible consiste à remplacer toutes les transitions de $\mathcal{D}^\infty(\mathcal{T})$ qui ne préservent pas le rang, i.e. $q' \xrightarrow{\sigma|u} q' \cdot \sigma$ avec $rg(q' \cdot \sigma) \subsetneq rg(q')$, par les transitions $q' \xrightarrow{\sigma|uv} \{(q, \epsilon)\}$ pour tout $(q, v) \in q' \cdot \sigma$. Le nouveau transducteur ainsi obtenu sera noté $\mathcal{W}^\infty(\mathcal{T})$.

On notera $\mathcal{W}(\mathcal{T})$ le transducteur obtenu à partir de $\mathcal{W}^\infty(\mathcal{T})$ en ne gardant que les états accessibles et co-accessibles. Il a été démontré dans [16] que si $[\![\mathcal{T}]\!]$ est multiséquentielle alors $\mathcal{W}(\mathcal{T})$ est un transducteur fini équivalent à $\mathcal{T}$.

De plus, $\mathcal{W}(\mathcal{T})$ possède un unique état initial et ses transitions non déterministes (i.e. telles qu'il existe une autre transition ayant le même état de départ et la même lettre d'entrée) sont toutes transitoires. On dit qu'il est *séparable*. On peut alors le décomposer en une union disjointe de transducteurs déterministes. Pour ce faire, on considère le graphe orienté $\Psi(\mathcal{W}(\mathcal{T}))$ dont les sommets sont les composantes fortement connexes de $\mathcal{W}(\mathcal{T})$ et dont les arêtes correspondent à ses transitions transitoires.

Pour tout chemin p dans $\Psi(\mathcal{W}(\mathcal{T}))$ commençant dans une composante fortement connexe contenant l'état initial de $\mathcal{W}(\mathcal{T})$, on note $\mathcal{W}(\mathcal{T})_p$ le transducteur obtenu à partir de $\mathcal{W}(\mathcal{T})$ en supprimant toutes les transitions transitoires n'apparaissant pas dans p. On peut alors décomposer $\mathcal{W}(\mathcal{T})$ comme la réunion disjointe des $\mathcal{W}(\mathcal{T})_p$ pour tous ces chemins p.

Bibliographie

[1] R. Alur et P. Cerný, « Streaming transducers for algorithmic verification of single-pass list-processing programs, » in *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, T. Ball et M. Sagiv, éd., ACM, 2011, p. 599-610. doi : 10.1145/1926385.1926454. adresse : https://doi.org/10.1145/1926385.1926454.

[2] R. Alur, L. D'Antoni, J. V. Deshmukh, M. Raghothaman et Y. Yuan, « Regular Functions and Cost Register Automata, » in *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, IEEE Computer Society, 2013, p. 13-22. doi : 10.1109/LICS.2013.65. adresse : https://doi.org/10.1109/LICS.2013.65.

[3] M. Béal et O. Carton, « Determinization of transducers over finite and infinite words, » *Theor. Comput. Sci.*, t. 289, n° 1, p. 225-251, 2002. doi : 10.1016/S0304-3975(01)00271-7. adresse : https://doi.org/10.1016/S0304-3975(01)00271-7.

[4] M. Béal, O. Carton, C. Prieur et J. Sakarovitch, « Squaring transducers : an efficient procedure for deciding functionality and sequentiality, » *Theor. Comput. Sci.*, t. 292, n° 1, p. 45-63, 2003. doi : 10.1016/S0304-3975(01)00214-6. adresse : https://doi.org/10.1016/S0304-3975(01)00214-6.

[5] J. Berstel, *Transductions and context-free languages* (Teubner Studienbücher : Informatik). Teubner, 1979, t. 38, isbn : 3519023407. adresse : https://www.worldcat.org/oclc/06364613.

[6] O. Carton, *Langages formels, calculabilité et complexité* (Capes-agrég). Vuibert, 2008, isbn : 978-2-7117-2077-4.

[7] C. Choffrut, « Minimizing subsequential transducers : a survey, » *Theor. Comput. Sci.*, t. 292, n° 1, p. 131-143, 2003. doi : 10.1016/S0304-3975(01)00219-5. adresse : https://doi.org/10.1016/S0304-3975(01)00219-5.

[8] C. Choffrut, « Une Caracterisation des Fonctions Sequentielles et des Fonctions Sous-Sequentielles en tant que Relations Rationnelles, » *Theor. Comput. Sci.*, t. 5, n° 3, p. 325-337, 1977. doi : 10.1016/0304-3975(77)90049-4. adresse : https://doi.org/10.1016/0304-3975(77)90049-4.

[9] C. Choffrut et M. P. Schützenberger, « Décomposition de Fonctions Rationnelles, » in *STACS 86, 3rd Annual Symposium on Theoretical Aspects of Computer Science, Orsay, France, January 16-18, 1986, Proceedings*, B. Monien et G. Vidal-Naquet, éd., sér. Lecture Notes in Computer Science, t. 210, Springer, 1986, p. 213-226. doi : 10.1007/3-540-16078-7_78. adresse : https://doi.org/10.1007/3-540-16078-7_78.

[10] L. Daviaud, I. Jecker, P. Reynier et D. Villevalois, « Degree of Sequentiality of Weighted Automata, » in *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, J. Esparza et A. S. Murawski, éd., sér. Lecture Notes in Computer Science, t. 10203, 2017, p. 215-230. doi : 10.1007/978-3-662-54458-7_13. adresse : https://doi.org/10.1007/978-3-662-54458-7_13.

[11] L. Daviaud, P. Reynier et J. Talbot, « A Generalised Twinning Property for Minimisation of Cost Register Automata, » in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, M. Grohe, E. Koskinen et N. Shankar, éd., ACM, 2016, p. 857-866. doi : 10.1145/2933575.2934549. adresse : https://doi.org/10.1145/2933575.2934549.

[12] S. Eilenberg, *Automata, languages, and machines. A* (Pure and applied mathematics). Academic Press, 1974, isbn : 0122340019. adresse : https://www.worldcat.org/oclc/310535248.

[13] E. Filiot, O. Gauwin et N. Lhote, « Logical and Algebraic Characterizations of Rational Transductions, » *CoRR*, t. abs/1705.03726, 2017. arXiv : 1705.03726. adresse : http://arxiv.org/abs/1705.03726.

[14] E. Filiot et P. Reynier, « Transducers, logic and algebra for functions of finite words, » *ACM SIGLOG News*, t. 3, n° 3, p. 4-19, 2016. adresse : https://dl.acm.org/citation.cfm?id=2984453.

[15] S. Gerdjikov, S. Mihov et K. U. Schulz, « A Simple Method for Building Bimachines from Functional Finite-State Transducers, » in *Implementation and Application of Automata - 22nd International Conference, CIAA 2017, Marne-la-Vallée, France, June 27-30, 2017, Proceedings*, A. Carayol et C. Nicaud, éd., sér. Lecture Notes in Computer Science, t. 10329, Springer, 2017, p. 113-125. doi : 10.1007/978-3-319-60134-2_10. adresse : https://doi.org/10.1007/978-3-319-60134-2_10.

[16] I. Jecker et E. Filiot, « Multi-Sequential Word Relations, » *CoRR*, t. abs/1504.03864, 2015. arXiv : 1504.03864. adresse : http://arxiv.org/abs/1504.03864.

[17] S. Mihov et K. U. Schulz, *Finite-State Techniques : Automata, Transducers and Bimachines* (Cambridge Tracts in Theoretical Computer Science). Cambridge University Press, 2019. doi : 10.1017/9781108756945.

[18] C. Reutenauer et M. P. Schützenberger, « Minimization of Rational Word Functions, » *SIAM J. Comput.*, t. 20, n° 4, p. 669-685, 1991. doi : 10.1137/0220042. adresse : https://doi.org/10.1137/0220042.

[19] M. P. Schützenberger, « A Remark on Finite Transducers, » *Inf. Control.*, t. 4, n° 2-3, p. 185-196, 1961. doi : 10.1016/S0019-9958(61)80006-5. adresse : https://doi.org/10.1016/S0019-9958(61)80006-5.